2013

Searching and describing human motion

Kevin Adistambha

University of Wollongong

Recommended Citation
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
School of Electrical, Computer, and Telecommunications Engineering

Searching and Describing Human Motion

Kevin Adistambha

Master of Engineering – Research

This thesis is presented as part of the requirement for the

Award of the Degree of Doctor of Philosophy

of the

University of Wollongong

February 2013
Thesis Certification

I, Kevin Adistambha, declare that this thesis, submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, is wholly my work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Kevin Adistambha
2013
Abstract

The amount of media being uploaded to the Internet is growing at an incredible rate. As an illustration, approximately 75 hours of video are uploaded to Youtube each minute, where approximately 30% of the videos contain human motion such as sport or music video. Consequently, new techniques and methods to search and describe contents related to human motion are sorely needed, since current search techniques mainly depend on user-supplied tags, which are often ambiguous and subjective when those tags are used to describe human motion. For example, a video containing “John Doe running and jumping into a lake” can be tagged as “John Doe”, “lake”, “running and jumping”, “funny video”, etc.

Being able to search for a specific motion has many applications. For example, searching for a specific movement in a sport in order to improve a person’s sporting performance by comparing to that of a professional athlete’s using automatically extracted movement features (such as a famous golfer’s swing, a famous tennis player’s forehand, etc.). This scenario will be possible if a method to objectively describe human motion existed. Searching human motion would be as natural as recording a motion and using it as yet another search term without having to think about the subjectivity of user-supplied tags and how someone else would “describe” that motion.

To achieve this, three things are required: a new multimedia communication format (since currently popular search techniques predominantly use simple text terms), a new human motion description language (since an objective and consistent method to describe human motion is also required), and feature extraction and matching technique for human motion search applications.

To communicate advanced multimedia queries, Multimedia Query Format (MQF) is presented in this thesis. MQF is a communication format for a structured multimedia search that goes beyond current text-based search currently in popular use. Instead of restricting itself to one particular multimedia description format, MQF was designed to allow the use of any number of current or future description standards, with advanced features for search such as logical operators, query-by-example, extensibility, and simplicity. MQF is also shown to work well with Fragment Request Unit (FRU) and Fragment Update Unit (FUU), which are MPEG standards
that enable selective synchronization of two XML documents over a network. Using FRU and FUU, MQF is shown to be able to perform “Query Streaming”, which is a continuously updatable multimedia query method that is suitable for use in mobile devices with limited resources. The work performed in MQF was also proposed to MPEG during the MPEG-7 Query Format standardization effort, where concepts introduced by MQF were contributed to the discussions, refinements, and validations during the MPEG standardization process.

To describe human motion objectively and accurately, Human Motion Markup Language (HMML) is presented in this thesis. HMML is a human motion description language that was designed to be able to describe human motion in three dimensions (sagittal, coronal, and transverse planes) to facilitate human motion centric search. Another design goal of HMML is to enable human motion search by utilizing MQF as the communication format, where HMML can be used in conjunction with existing multimedia description standard such as MPEG-7 and Dublin Core to provide a more complete description of a desired media not currently possible today. Key features of HMML includes human readability, simplicity, and searchability.

To extract this objective human motion description, a method to automatically extract HMML motion description from 3D motion capture data is also presented. This method involves “partial reconstruction” of the human body, i.e., each of the major limb such as the arms and the legs are reconstructed from 3D data independently. By not reconstructing the body as a whole, each limb becomes a separate entity that can be described independent of other limbs in an objective manner. Consequently, applications searching for a walking motion with the leg movements to serve as the query term will also match walking and waving, walking and dribbling, etc., providing a fine-grained method for motion search. Experiments were performed to determine the consistency of the extracted symbol sequences using walking, running, and sneaking motions, where it was found that the extracted symbols are consistent even when the symbols were extracted from people of varying height and movement patterns. Also, the optimal motion duration and detail level of the extracted symbol sequence were investigated to utilize the symbol sequences in a motion matching application.
Acknowledgments

I would like to thank my family for their interminable support during the long hours of work that resulted in this thesis. This is simply not possible without their support.

I would also like to thank my supervisors Christian H. Ritz, Ian S. Burnett, and David Stirling for their guidance and suggestions.

Also thank you to Stephen J. Davis and all the people in the lab (Muawiyath Shujau, Mohammed Barakat, Xiguang Zheng) for the discussions and regular coffee sessions throughout the years.
Contents

1 Introduction .......................... 1
  1.1 Multimedia Search and Human Motion .......................... 1
  1.2 Human Motion Search and Description .......................... 3
  1.3 Thesis Outline .......................... 5
  1.4 Contributions .......................... 5
  1.5 Publications .......................... 6
    1.5.1 Conference Publications .......................... 6
    1.5.2 Journal Publication .......................... 7
    1.5.3 MPEG Input Document .......................... 7

2 Multimedia Metadata and Human Motion .......................... 8
  2.1 Introduction .......................... 8
    2.1.1 Multimedia and Metadata .......................... 8
    2.1.2 Human Motion .......................... 9
  2.2 Describing Multimedia .......................... 10
    2.2.1 Binary-based Descriptions: ID3 .......................... 11
    2.2.2 Structured Textual Descriptions .......................... 13
    2.2.3 RDF-based Multimedia Descriptions: Dublin Core .......................... 25
    2.2.4 XML-based Multimedia Descriptions: MPEG-7 .......................... 27
  2.3 Searching Multimedia .......................... 28
    2.3.1 Web-based Technologies .......................... 29
    2.3.2 Query-by-example .......................... 30
    2.3.3 Fingerprinting .......................... 30
    2.3.4 Human Motion Oriented Search and Classification .......................... 31
    2.3.5 Multimedia Search and Description: Summary .......................... 32
  2.4 The Human Body, Motions, and Descriptions .......................... 33
    2.4.1 Human Perception: Point Light Analysis .......................... 33
    2.4.2 Anatomical Description of the Body .......................... 35
    2.4.3 Text-based Descriptions .......................... 38
    2.4.4 Notation-based Descriptions .......................... 39
    2.4.5 Motion Capture .......................... 40
CONTENTS

4.3.1 Requirements .................................................. 105
4.3.2 The HMML Schema Design ................................. 107
4.3.3 The root <hmml> Element ................................. 108
4.3.4 The <sagittal>, <coronal> and <transverse> elements .... 108
4.3.5 The <motion> element ................................. 112
4.3.6 The <repeat> element ...................................... 117
4.4 Semantic Motion Query Using HMML ....................... 119
4.4.1 The HMML description of a walking Motion ............... 119
4.4.2 Using HMML description fragments to form a temporal query 119
4.5 HMML Query Using MQF ............................................ 127
4.5.1 Single Server Query Scenario .......................... 127
4.5.2 Multi Server Query Scenario .......................... 129
4.6 HMML Size Comparison Experiments ......................... 132
4.6.1 Methodology ............................................. 134
4.6.2 Results .................................................. 135
4.7 Conclusions .................................................... 137

5 Motion Feature Extraction .......................... 139
5.1 Introduction ................................................... 139
5.2 Extracting Features From Motion Capture Data ............... 140
5.2.1 Partial Reconstruction of the Motion Capture Skeleton Hier-
archy ............................................................. 142
5.2.2 Determining the Angular Values .......................... 147
5.2.3 Symbolization of the Angular Values ....................... 149
5.2.4 Movement Detection Threshold (MDT) .................... 156
5.2.5 Symbol Sequence Uniqueness ............................ 161
5.3 Symbol Sequence Timing Analysis .......................... 164
5.3.1 Filtering The Symbol Sequences .......................... 164
5.3.2 Symbol Sequence Consistency For Walking Motion ........ 166
5.3.3 Motions With Identical Filtered Sequences ............... 169
5.3.4 Discussion ............................................. 175
5.4 Symbol Sequence Use Cases and Applications ................. 176
5.4.1 Automatic Annotation of Motion Capture Data using HMML 176
5.4.2 Motion Matching ........................................... 178
5.5 Conclusions .................................................... 188

6 Conclusions and Future Work .................... 190
6.1 Conclusions ................................................... 190
6.2 Future Works .................................................. 191
CONTENTS

Bibliography 194

A Multimedia Query Format XML Schema 203
B Human Motion Markup Language XML Schema 208
C The Echoprint Algorithm 215
D The BVH Format 218
E MQF Audio Query Example Instantiations 224
F Example of an HMML Authoring Tool 228
List of Figures

1.1.1 Overview of the three parts of the thesis: multimedia query communication; description of human motion; and feature extraction of human motion from motion capture data. .......................... 1

1.2.1 Illustration of the problem of describing human motion in a semantic, searchable manner. ................................................................. 4

2.2.1 Visualization of the ID3 Version 2 container (in yellow) and the frames that contained information about the media within (in blue); the ID3 Version 2 container is prepended to the content (in white). .......................... 12

2.2.2 John’s online store in HTML format. <h1> denotes a heading, <p> denotes paragraph, and <b> denotes boldface font. ............................ 14

2.2.3 John’s online store HTML formatting from Figure 2.2.2 rendered in a web browser. ................................................................. 15

2.2.4 An example of XML representation of John’s Online Store. ............. 16

2.2.5 XSLT code used to transform the XML of Figure 2.2.4 into Figure 2.2.2. ................................................................. 17

2.2.6 Block diagram showing FRUs and FUUs to retrieve fragments of XML from a remote server. .................................................. 18

2.2.7 An example Fragment Request Unit (FRU) requesting a part of a document called “MPEG7Description.xml”. The FRU is requesting the node called “Classification”. ................................. 20

2.2.8 An example reply from the server using Fragment Update Unit (FUU). The FUU describes that the requested “Classification” node exists in the path of “/Mpeg7/Description/CreationInformation/Classification” and returns the contents of the node. ................................. 20

2.2.9 An example of RDF graph describing Dr. Eric Miller, his email address, full name, and title (example taken from [25]). ...................... 22

2.2.10 The relationship between Uniform Resource Identifier (URI), Uniform Resource Location (URL), and Uniform Resource Name (URN). Both URL and URN are subsets of URI. ................................. 24

2.2.11 XML-based representation of the RDF graph in Figure 2.2.9 [25]. ... 24
LIST OF FIGURES

2.2.12 Scope of MPEG-7 (in yellow) [16]. ........................................... 27
2.2.13 Elements of MPEG-7 defined in the standard (in yellow), description elements (in blue), structural elements of a description (in orange), and XML instantiation of MPEG-7 descriptions (in white) [16]. ... 27
2.2.14 An example instantiation of MPEG-7 description fragment describing the Jazz genre of a media. ......................................................... 28
2.3.1 Image query by example. ............................................................ 30
2.3.2 An example of Echoprint feature string. The features are arranged in feature-timestamp format. ......................................................... 31
2.3.3 An example Echoprint codebook, storing fingerprints of all known audio files. ................................................................. 31
2.4.1 Illustration of a Point Light Analysis of human movement. In this example, the points are located in the major features of the body; the head, the shoulders, the elbows, the wrist, the chest, the hips, the knees, and the foot. ..................................................... 34
2.4.2 The planes dividing the human body: The Sagittal plane describes movement in forward-backward directions, the Coronal plane describes movement in left-right directions, and the Transverse plane describes up-down movements (Graphics by Mrabet [76] Licensed under Creative Commons Attribution and ShareAlike license (CC-BY-SA)). ... 36
2.4.3 Right leg coronal, transverse, and sagittal plane anatomical movements viewed from the front and the side. ......................................... 37
2.4.4 Labanotation symbols and staff [78]. The dotted line in the staff is not drawn. The notation is read from bottom to top. ......................... 41
2.4.5 Example of Labanotation for the motion scenario in Section 2.4.3: “The person takes four steps starting with the right leg, where the steps are alternating between small steps and large steps, and takes four steps backward of normal length using a similar timing to the normal forward steps. The hands are moving in sync to the same direction of the same side’s legs. The motion starts and ends with the body standing straight.”. Labanotation is read from bottom to top; where part (b) is a continuation of part (a). ......................... 42
2.4.6 Illustration of motion capture of a walking motion. Although similar to point light analysis of human movement (shown in Figure 2.4.1), motion capture utilizes of the knowledge of the skeletal structure of the body. The point lights are connected by lines that show the bones. 43
2.4.7 An illustration of a closed kinematic chain connecting three rigid segments using three joints. The movements of the joint angles $\alpha$, $\beta$, and $\gamma$ are interconnected, and changes in one of the angles affects the others. .......................................................... 45

2.4.8 An illustration of an open kinematic chain connecting three rigid segments and an end effector using three joints. The angles $\alpha$, $\beta$, and $\gamma$ are required to manipulate a robotic arm with known segment lengths so that the end of the arm (i.e. the end effector) is located at a desired coordinate with respect to the origin (i.e. the base of the arm). A change in one of the joint angles does not affect the others, unlike in a closed kinematic chain. [95] ........................................... 46

2.4.9 Human motion illustration of (A) open kinematic chain of arm waving, and (B) closed kinematic chain of a squat motion. ................. 47

2.4.10 A simplified version of Figure 2.4.8 showing the Inverse Kinematics process on a 2D plane. The two segments are S1 and S2, with an end-effector E located at positions X and Y. The goal of Inverse Kinematics is to calculate the angles $\sigma$ and $\beta$. ................. 47

2.4.11 Typical motion capture representation of the human body. ........ 49

2.4.12 Typical motion capture skeleton hierarchy. The hierarchy shown in this Figure is a representation of the skeleton shown in Figure 2.4.11. 50

2.4.13 A simplified 2D illustration of global and local spaces that describes the movements of a bone. This example illustrates the movement of a bone described by the vector $\overrightarrow{L_XP}$ on a 2D plane, where $P$ is a point defined in the local space at coordinates $(X_{local}, Y_{local})$, $L_X$ and $L_Y$ describes the position of the local space in terms of the global space, and the local space is rotated (i.e. moved) $\theta$ degrees with respect to the global space. The goal of Forward Kinematics is to calculate the position of $P$ in terms of the global space coordinates. ................. 52

3.0.1 The context of the Chapter in terms of the overall system: query communication (highlighted). .......................... 59

3.1.1 The general concept of a multi purpose multimedia query format, enabling a single query to be used to search large centralized multimedia databases and small personal multimedia databases. ................. 62

3.4.1 The root <mqf> element illustration. .......................... 70
3.4.2 Example MQF instantiation of the root <mqf> element that contains the elements <queryId> and <from>, and attributes of “version” and “xml:lang”. Note that this instantiation is not yet well-formed since the required sub-elements (one of <query>, <replies>, <exceptions> or <services>) are not present. 71

3.4.3 An illustration of the use of <item> and <operator> elements inside the <query> element to specify the arithmetic process of (1+2)*3 in MQF. 72

3.4.4 The <query> element illustration. 73

3.4.5 The <item> element illustration. 75

3.4.6 The <replies> element illustration. The element encapsulates all individual match items in the result set, each in a separate <reply> element. A result set can also contain a general description of the set in a textual format, which is encapsulated in a <text> element. 76

3.4.7 The <reply> element illustration. 77

3.4.8 The <description> element illustration. 77

3.4.9 The <services> element illustration. 78

3.4.10 The <exception> element definition which contains the exception code and textual description of the error. The <code> element contains an HTTP-compatible error codes along with a textual description of the error inside the <description> element. 80

3.4.11 An example instantiation of a server exception using HTTP code 503 (service unavailable). This error could signify that the server is down for maintenance. 80

3.5.1 A typical sequence of query communication using MQF. 81

3.5.2 The initial service discovery step. The client initiates the connection, and service discovery has to be performed before any query can proceed. 82

3.5.3 The server’s service discovery reply showing the query level capabilities of the server, media types available in the database, and allowed logical operators. This example shows that the server is capable of performing exact match and query-by-example (but not semantic nor freetext queries), contains images (in JPEG format), audio (in MP3 format), and video (in MP4 format), and supports AND, OR, and NOT operations. The presence of Dublin Core description namespace (“http://purl.org/dc/elements/1.1/”) signifies that the server’s contents are described using Dublin Core. 83

3.5.4 The client performs a query of “search for a video where John Doe is running and jumping into a lake” using Dublin Core descriptions as part of the query terms. 84
3.5.5 The server replies with two relevant matches in its database. .......... 85
3.5.6 If the client’s request is not well-formed XML or unrecognizable by
the server, the server will reply with HTTP-compatible status code.
In this example, the status code returned is 400 (Bad request) due to
the error in the client’s formulation of the query. ................. 86
3.7.1 An illustration of a query-by-example query refinement process. .... 89
3.7.2 Workflow of stateful MQF multimedia query using Query Streaming.
Both the client and the server possess a copy of the client’s query and
the result set. Both documents are synchronized using FRU and FUU. 91
3.7.3 Image query scenario using Query Streaming. The client requested
pictures of a mountain (1), where there are 20 matching images with
the keyword “mountain” in the server (2). The client instructs the
server to provide additional keywords associated with those pictures
(3) and the server sends them to the client (4). The client adds an
additional term of “snow” to the original query (5), and the server
replies that there are only three pictures that uses both “mountain”
and “snow” keywords (6). ............................................ 93
3.7.4 Audio query scenario using Query Streaming. The client constructs
an initial query using an attached snippet of an audio (1), where the
server finds 100 items that is similar to the client’s example (2). The
client asks for the genres associated with those 100 items (3), and the
server replies with the genres of “rock”, “country”, and “classical” (4).
The client then updates the original query with an additional term
of “genre is country” (5), and the server reports that there is one file
matching the query (6). ............................................ 96
4.0.1 The context of the chapter in terms of the overall system: the de-
scription of human motion (highlighted). .......................... 101
4.3.1 Illustration of the root element of HMML. ........................... 109
4.3.2 The primary building blocks of the HMML motion description: (a)
the planar element (only the <sagittal> element is shown for brevity)
which contains the limb movement description elements shown in (b)
where only the <leg> element is shown for brevity. For both elements,
the other planar elements (<coronal>, <transverse>), and the other
limb elements (<arm>, <torso>, <head>) are defined similarly. .... 110
4.3.3 An example HMML instantiation describing walking forward four
steps starting with the left leg, described in the sagittal plane. .... 111
4.3.4 The <motion> element graphical representation. ............... 112
4.3.5 An example HMML instantiation of a logically grouped motion. In this case, walk and run. ................................................ 114
4.3.6 Ambiguous order of leg movements without the use of the “start” and “end” timing attributes, where the four leg movement descriptions have no obvious order. ................................. 114
4.3.7 Removing ambiguous interpretation of movement occurrence order using motion grouping. ................................. 115
4.3.8 High-level description of a motion using only <motion> elements to describe a walking motion followed by a running motion. This example is a more general representation of the motion described in Figure 4.3.5. ........................................ 116
4.3.9 The <repeat> element graphical representation. ......................... 117
4.3.10 An example HMML instantiation of a repeated motion. The example shows two steps forward repeated twice, resulting in walking four steps forward. Each of the leg-forward motion would need to be encapsulated inside a <motion> element to remove ambiguity in movement order as described in Section 4.3.5 .................................... 118
4.4.1 An illustration diagram of HMML describing leg and arm movements temporally in the sagittal plane. ................................. 120
4.4.2 HMML instantiation in XML for the diagram shown in Figure 4.5.1. This document is assumed to exist in the server. ......................... 121
4.4.3 An illustration of HMML temporal matching example by matching the incoming query to a subset of an existing description. The incoming query (labeled as “Query” in the diagram) matches part of the description stored in the database (labeled as “In Database” in the diagram). ........................................ 122
4.4.4 Example XML of an HMML fragment forming a temporal query (labeled as “Query” in Figure 4.4.3). ................................. 122
4.4.5 An illustration of matching repeating group using the <repeat> element to match walk four steps by using walking two steps repeated twice. An example XML instantiation of the Query portion is shown in Figure 4.4.6. ................................. 124
4.4.6 The XML instantiation of the query portion of the diagram shown in Figure 4.4.5. Each of the leg-forward motion would need to be encapsulated inside a <motion> element to remove ambiguity in movement order as described in Section 4.3.5. ................................. 125
4.4.7 Example XML instantiation of a motion tagged by its semantic content using textual tags. In this case, “walk”, “run”, and “jump”. Each tagged content specifies the starting and ending frame of the sub-motion. Textual search using keywords of “walk”, “run”, and/or “jump” will return this document. 125

4.4.8 An illustration of using HMML in conjunction with Dublin Core and MPEG-7 to perform a detailed search of a video using the query terms as an increasingly strict filtering process. 126

4.5.1 An example scenario of a multimedia query using HMML transported in MQF. 128

4.5.2 MQF-encapsulated HMML search representing the query portion of the diagram shown in Figure 4.5.1. 129

4.5.3 MQF-encapsulated HMML search representing the query portion of the diagram shown in Figure 4.5.1. 130

4.5.4 MQF-encapsulated HMML query reply (using the example shown in Figure 4.5.1). Metadata returned included Dublin Core and MPEG-7. 131

4.5.5 Querying Human Motion Description using MQF: multiple server query scenario. 132

4.5.6 XML representation of aggregated MQF encapsulated query reply from two multimedia databases. 133

4.6.1 A typical example of HMML annotation of a walking forward four steps motion. 135

5.0.1 The context of the chapter in terms of the overall system: motion feature extraction (highlighted). 139

5.2.1 Block diagram of the automatic motion semantics extraction process. Limbs are extracted from the source media, where the movements in the sagittal, coronal, and transverse planes (in the Plane Separation process) for each limb are detected. The moments between movements are assigned predefined symbols, and the flow from one symbol to the next are recognized as the motion semantics. 141

5.2.2 An illustration of the problem of body facing in feature extraction of motion capture recording. (a) the reconstructed motion shows a person walking in a semi-circle to the left, and (b) normalized motion reconstruction so that the person is facing a constant direction. 143

5.2.3 Visualization of the bones that participate in partial reconstruction (in yellow) and bones that do not (in white) in the typical motion capture skeleton hierarchy. 145

5.2.4 Partially reconstructed left leg and its extremity. 146
5.2.5 Projection of a limb represented by the vector \( \overrightarrow{OP} \) into its sagittal (the angle \( \alpha \)) and coronal (the angle \( \beta \)) plane components. The transverse plane component is represented by the length of the vector \( \overrightarrow{OP} \). ... 148

5.2.6 3D view of a leg unit projected into the sagittal and coronal plane. Point O represents the origin of the limb unit (in this case the hip) and point E represents the extremity of the limb unit (in this case the foot). The limb unit is projected into the two anatomical planes (sagittal and coronal). ... 150

5.2.7 An example angular value plot of a walking motion after partial reconstruction of the left and right legs. The subject is walking five steps forward starting with the right leg. ... 151

5.2.8 Correlation between a plot of reconstructed angular values in a walking motion to the sub-motion of walking. In this example, the two sub-motions are one forward step with the left leg and one forward step with the right leg, where a “forward step” is defined from the moment where the leg swinging forward up to the moment where it makes non-sliding contact with the ground to receive the body weight. 152

5.2.9 Symbolization of the peak and throughs of the angular plot for the legs for a walking motion in the sagittal plane. The left leg peak is symbolized as \( a \); the left leg through is symbolized as \( b \); the right leg peak is symbolized as \( c \); and the right leg through is symbolized as \( d \). 152

5.2.10 Detailed workflow of the symbol stream extraction process in the sagittal plane of a walking motion. The resulting symbol stream in this example is \( c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c \). Specifically, the symbol stream generated is \( c-d-a-b-c-d-a-b-c \) and \( n-o-p-m-n-o-p-m \) for the legs and the arms, respectively. ... 155

5.2.11 An illustration of the use of the Movement Detection Threshold (MDT) value. If the MDT value is set at 0, then points \( x \), \( y \), and \( z \) are all recognized as valid turning points. However if the MDT value is set to 2, point \( y \) would not be recognized as a valid turning point since the difference between \( x \) and \( y \) is less than the MDT value, but point \( z \) will be recognized since the difference between point \( z \) and a previous turning point (even if it’s not included in the list of valid turning points, which is point \( y \) in this case) is more than the MDT value. ... 157

5.2.12 The effect of different MDT values on a fragment of extracted symbol sequences for a walking motion. ... 158
5.2.13 The average number of features extracted per file for MDT values of 0-20 for the HDM motion capture database. Total number of files is 270, where the average length of each file is 41 seconds. Error bars in the graph represent the 95% confidence interval.  

5.2.14 Plots showing the precision and recall values of the matching algorithm using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x axis shows the MDT value of both the input symbol sequence and the codebook symbol sequence was extracted at. Error bars represents 95% confidence intervals.  

5.3.1 Result of an experiment to examine the consistency of symbol sequences extracted from walking motion of four subjects of different heights and different walking patterns. The extracted symbol sequences for the four subjects show a consistent pattern in the symbol sequence of the leg movements in the sagittal plane (b-c-d-a, highlighted in red).  

5.3.2 Timing analysis of each symbol occurrence in the b-c-d-a pattern in a walking motion shown in Figure 5.3.1. The timings of the symbol occurrences in walking motion show a relatively consistent between subjects of different heights.  

5.3.3 The timings of symbol occurrences in walking motion from three different databases. The symbol b serves as the starting point of time measurements to the other symbols, hence the time that b occurred is always zero.  

5.3.4 Running motion analysis from two different databases. The symbol b serves as the starting point of time measurements to the other symbols, hence the time that b occurred is always zero.  

5.3.5 Sneaking motion analysis. The symbol b serves as the starting point of time measurements to the other symbols, hence the time that b occurred is always zero.  

5.3.6 Average symbol timings between three types of motion that generates identical symbol sequences (b-c-d-a). The error bars which represent 95% confidence interval do not overlap between the three types of motions having the same symbol sequence, showing a statistical difference between the symbol timings of the three motions. Therefore, using the timing information in combination with the symbols, motions that generate identical symbol stream can still be differentiated from one another.  

5.4.1 The overall process of HMML extraction from motion data.
5.4.2 The source angular plot from which automatic HMML annotation is performed. .......................... 179

5.4.3 An example of an HMML document derived for a walking motion. This example shows the left leg stepping forward from frames 55 to 132, the right leg stepping forward from frames 132 to 207, the left leg stepping forward from frames 207 to 283, and the right leg stepping forward from frames 283 to 360. ................................. 179

5.4.4 A method to match a known symbol sequence to an input symbol sequence for motion. This matching technique is used in the Echoprint music matching algorithm [9] and was utilized for this experiment. The steps involved are: (1) The input symbol sequence; (2) Identification of relevant frame-symbol pairs; (3) Calculation of the time differences between symbols; and (4) Comparison with a known sequence in the database. .................................................. 183

5.4.5 Plots showing the precision and recall values of the high-detail motion database experiment using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x axis denotes the MDT value used in the extraction process for the input sequence, and the motion database was extracted using an MDT value of 0. Error bars represent 95% confidence intervals. ......................... 185

5.4.6 Plots showing the precision and recall values of the low-detail motion database experiment using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x axis denotes the MDT value used in the extraction process for the input sequence, and the motion database was extracted using an MDT value of 20. Error bars represent 95% confidence intervals. ......................... 186

C.0.1 Echoprint algorithm initial scoring: The initial score is calculated for each entry in the codebook based on the features that are present in both the input query and the codebook entries. In this example, codebook “Entry 1” initial score is 2, and codebook “Entry 2” initial score is 1. ................................................................. 216

C.0.2 Echoprint algorithm delay calculation: For each matching features in the input query and the codebook, the timestamp differences are calculated. In this example, in Entry 1, the delay for both features “1024810” and “1027459” are 10. In Entry 2, the delay for feature “1024810” is -20. ................................................................. 216
C.0.3 Echoprint algorithm final score calculation: For each delay value calculated from Figure C.0.2, identical scores are counted. In this example, Entry 1 has two delays of 10, which results in a final score of 2 (two counts of delays of 10). For Entry 2, there is only one delay value (which is -20), therefore the final score for Entry 2 is 1 (one count of delay of -20).

D.0.1 An example of a BVH file fragment showing the hierarchy and motion of the hip, left hip join, and left femur bones for one frame from the file 35_01.bvh (available from [96]).

D.0.2 BVH example: the root bone as defined in the skeleton hierarchy and its corresponding motion for the first frame.

D.0.3 BVH example: the left hip joint and its corresponding motion for the first frame.

D.0.4 BVH example: the left femur and its corresponding motion for the first frame.

E.0.1 Updateable audio query step 1: The client sends an audio example using an attached binary sample encoded in Base64. The query level is set to query-by-example.

E.0.2 Updateable audio query step 2: The server replies that there are 100 possible matches found that is similar to the sample given by the client.

E.0.3 Updateable audio query step 2a: The result set that is kept in the server. This document is not sent to the client.

E.0.4 Updateable audio query step 3: The client’s FRU request to retrieve the genres in the result set shown in Figure E.0.3. Using FRU, the client does not need to receive the whole result set, however, the client can perform additional examination of the result set.

E.0.5 Updateable audio query step 4: The FUU replies from the FRU of Figure E.0.4. The server replied with the genre description of the result set that is kept in the server.

E.0.6 Updateable audio query step 5: The additional term to be added to the initial query in Figure E.0.1. The client requested that the server transmits items in the result set that belongs to the “Country” genre.

E.0.7 Updateable audio query: The final query as seen by the server – the client requests that the previous result set (arrow 1) to be filtered to only include songs of the genre country (arrow 2).

F.0.1 Example of an HMML authoring tool.
List of Tables

2.2.1 Dublin Core version 1.1 descriptors (source: [49]). ............................ 26
3.6.1 The performance measurement results of MQF using a variety of query scenarios. ................................................................. 88
3.7.1 Image query scenario without Query Streaming, uncompressed. .... 94
3.7.2 Image query scenario without Query Streaming, BiM compressed. ... 94
3.7.3 Image query scenario with Query Streaming, uncompressed. .......... 95
3.7.4 Image query scenario with Query Streaming, BiM compressed. ...... 95
3.7.5 Audio query scenario XML files, uncompressed. .......................... 98
3.7.6 Audio query scenario XML files, BiM compressed. ....................... 98
4.6.1 The number of files and the motion types used in the size testing. ... 134
4.6.2 Average uncompressed and compressed size of the motion files in BVH format used in the HMML annotation testing, grouped by motion types. 135
4.6.3 Average HMML element count and HMML file sizes achieved by annotating the files shown in Table 4.6.2. On average across four motion classes and 56 motion files, the HMML size (in Bytes) are 99.6% smaller compared to the size of the original BVH files. ...................... 136
4.6.4 Size comparisons of gzip-compressed HMML and BiM compressed HMML against uncompressed HMML, and uncompressed HMML and BiM-compressed HMML against original uncompressed BVH sizes. ... 136
5.2.1 Limb makeup of motion capture partial reconstruction. ................. 146
5.2.2 Correlation between feature symbols and limb movements (arms and legs). ................................................................. 153
5.2.3 Correlation between feature symbols and limb movements (backbone and head). Note that the symbols associated with the turning points are case-sensitive. ......................................................... 154
5.2.4 The total number of symbols extracted from the whole HDM motion capture database using different MDT values. ....................... 159
5.3.1 Examples of symbol sequences extracted from walking forward, walking backward, sneaking, and running motions. The extracted symbols contain symbols from all limbs and all anatomical planes. Filtering the extracted symbols to remove all symbols except the ones that represent the leg movements in the sagittal plane reveals the similarities between walking, sneaking, and running motions: the sequence \( b-c-d-a \). The two walking backward motions contain the sequence \( c-b-a-d \) instead.

5.4.1 The partial translation list (for brevity) showing the correlation between the symbol sequences and its HMML description counterpart.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC</td>
<td>Acclaim Motion Capture</td>
</tr>
<tr>
<td>ASF</td>
<td>Acclaim Skeleton File</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BVH</td>
<td>Biovision Hierarchy</td>
</tr>
<tr>
<td>BiM</td>
<td>Binary MPEG Format for XML</td>
</tr>
<tr>
<td>CMU</td>
<td>Carnegie Mellon University</td>
</tr>
<tr>
<td>DC</td>
<td>Dublin Core</td>
</tr>
<tr>
<td>DCMI</td>
<td>Dublin Core Metadata Initiative</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>FRU</td>
<td>Fragment Request Unit</td>
</tr>
<tr>
<td>FUU</td>
<td>Fragment Update Unit</td>
</tr>
<tr>
<td>HDM</td>
<td>Hochschule der Medien</td>
</tr>
<tr>
<td>HMML</td>
<td>Human Motion Markup Language</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>ISBN</td>
<td>International Standard Book Number</td>
</tr>
<tr>
<td>LMA</td>
<td>Laban Movement Analysis</td>
</tr>
<tr>
<td>MDT</td>
<td>Movement Detection Threshold</td>
</tr>
</tbody>
</table>
MP3  MPEG-I Layer III Audio Compression
MPEG  Moving Pictures Expert Group
MP7QF MPEG-7 Query Format
MPQF  MPEG Query Format
MQF   Multimedia Query Format
RDF   Resource Description Framework
RSS   Really Simple Syndication
SGML  Standard Generalized Markup Language
SOAP  Simple Object Access Protocol
UOW   University of Wollongong
URI   Uniform Resource Identifier
URL   Uniform Resource Locator
URN   Uniform Resource Name
W3C   World Wide Web Consortium
WSDL  Web Services Description Language
WWW   World Wide Web
XHTML Extensible Hypertext Markup Language
XML   Extensible Markup Language
XMPP  Extensible Messaging and Presence Protocol
XSLT  Extensible Stylesheet Language Transformations
Chapter 1

Introduction

1.1 Multimedia Search and Human Motion

The amount of user-generated multimedia content uploaded into video sharing sites such as Youtube [1], Vimeo [2] or Dailymotion [3] is putting an increasing pressure on search technologies to enable a user to find a desired media quickly and efficiently. The latest statistics published by Youtube in 2012 [4] show that there are 72 hours of videos uploaded every minute, with 800 million viewers per month. The statistics
also show that there are 3 hours of video uploaded every minute from mobile devices, and 20% of the traffic to Youtube comes from mobile devices.

Currently, multimedia search technologies depend on the use of user-supplied keywords (“tags”) which are provided by the uploader. Due to its subjective nature, tags are potentially ambiguous, and may or may not describe the actual content of the media being uploaded. For example, a video containing “John Doe running and jumping into a lake” may be tagged as: “John Doe”, “hilarious”, “vacation”, “lake”, etc. The same video uploaded by another person may be tagged as “family”, “road trip”, “2012”, etc. The tags are therefore reflecting the subjective context of the video according to the point of view of the uploader. Tagging experiments performed by Davis et al. [5] also found that the background of the person supplying the tags influences the tags themselves, with noticeably different tags being supplied by people from technical (i.e. engineers, scientists) and non-technical background.

To describe the content of the John Doe video objectively (i.e. describing the actual content of the video instead of describing a single person’s opinion of the content), a method other than tagging is therefore needed. Such objective description method exists in at least two forms: MPEG-7 Multimedia Description Scheme [6], and Dublin Core Metadata Initiative [7]. MPEG-7 provides methods to describe a media content using the objective features of the media (e.g. color, spatial information, etc.), while Dublin Core focuses on user-supplied information that are as non-subjective as possible (e.g. author name, publisher information, title, etc.).

Unfortunately, both MPEG-7 and Dublin Core description standards do not define how a human motion should be described. In the video of “John Doe running and jumping into a lake”, both MPEG-7 and Dublin Core provide description methods for John Doe (e.g. subject is John Doe) and lake (e.g. the color of a section of the video is blue), but the description of the act of “running and jumping” currently would have to rely on tag-based descriptions.

A survey conducted by Cheng et al. [8] using 3,269,030 video uploaded to Youtube reveals that 22.9% of the video uploaded to Youtube were categorized as “music” (many of which potentially contain dance motions), and 9.5% of the video uploaded were categorized as “sport” (which also contain human motion). Being able to search the videos by motion would therefore help refine the search results on one-third of the videos available in Youtube [4].

Therefore, searching human motion in multimedia is a problem that requires new solutions to be created (depicted in Figure 1.1.1), which includes: a new searchable description format to describe human motion; a new communication protocol to transmit those human motion descriptions; and a method to extract the human motion descriptions automatically.
CHAPTER 1. INTRODUCTION

1.2 Human Motion Search and Description

Despite the fact that sophisticated multimedia description schemes such as MPEG-7 exists, the typical method to describe when a person performs a motion in a video is to manually segment the motion of interest and upload that segmented clip, and describe that clip using tags and textual descriptions. There are certain problems with this “manual segment-and-describe” approach; particularly, there is no reliable method to semantically search a motion performed in a video. For example, consider the following scenario:

1. John and his friends went to a lake, where a video was being recorded continuously.

2. At some point in the recording, John ran and jumped into the lake.

3. The scenario was found to be funny by John’s friends, and his friends wanted to show everyone what John did.

After the trip, it was decided to upload the video, and highlight the “John running and jumping into the lake” scenario. Immediately, there are several obstacles:

1. The act of John running and jumping into the lake only occurred on a short, specific section of the whole recording.

2. His friends would have to manually segment and tag the video of the exact event, and upload it separately from the original video where it was coming from (thus creating redundancy).

3. “Running and jumping into a lake” is a highly subjective and ambiguous description of what John actually did that his friends found funny.

4. Even if John’s friends are knowledgeable in the area of multimedia descriptions, currently available multimedia description standards does not provide a standardized method to describe human motion. Therefore they had to resort to basic tags and textual descriptions.

An example of the problem from the lack of a standardized human motion description is depicted in Figure 1.2.1: current tagging approach allows the description of a video clip as a whole, where in Figure 1.2.1 the clip is titled “walking four steps to the right”, and the associated tag is “walking”. Although the title provides sufficient description of the motion being performed, the tag (i.e. walking) by itself does not provide sufficient detail about the motion. Using a tagging approach, more tags will be needed, such as “walking”, “four steps”, “normal step size”, “walking to the right”, etc. However, the tags are describing the video as a whole. If a user would
CHAPTER 1. INTRODUCTION

Tag: walking
Description: walking four steps to the right

Figure 1.2.1: Illustration of the problem of describing human motion in a semantic, searchable manner.

like to tag a specific part of the video such as “the moment where the right leg is stepping forward”, global tags does not provide a solution. Temporal tags partially solves the problem of describing moments, but the complexity of human motion will result in a multitude of tags that will quickly overwhelm the user. For example, the moment where the right leg steps forward in 1.2.1 (which consists of 116 frames) could be tagged as “one step”, “right leg”, “forward”. The lack of context in the tags presented another difficulty: what does “forward” mean? Does “forward” describe the right leg movement or the whole body moving forward? A human motion description that is temporal, objective, and accurately describes the motion being performed is therefore a requirement for multimedia search using human motion.

With the renewed attention toward motion capture lately with the availability of motion-enhanced gaming devices such as Nintendo Wii, Playstation Move, and Microsoft Kinect, the motion description problem discussed above also applies. While motion capture technology has been available for some time for use in movies and research purposes, describing the content of a motion capture data is an open research question. Arguably, a standardized motion description is more vital for motion capture data since unlike a video, a motion capture data has no other visible context that is present in the recording besides the motion itself.

This thesis would therefore propose a solution for each of the problem depicted in Figure 1.1.1 and Figure 1.2.1: the transport aspect of a human motion centric multimedia query; the description of human motion that is designed for multimedia search by working in conjunction with existing multimedia metadata description
standards; and automatic extraction of human motion description from motion capture data, along with an analysis on the consistency and searchability of the human motion description extracted from people with different heights.

A vertical approach was chosen for the work presented in this thesis due to the fact that there are minimal work that combines all three aspects required to perform a human motion search (i.e. query protocol, human motion description, and motion description extraction). Although individually the three aspects are complex problems on its own, it is difficult to conclude that human motion search is feasible without the three aspects shown to be feasible individually. For example, by developing a query protocol for human motion without a feasible human motion description, the conclusion that human motion search is possible cannot be reached. This thesis thus focuses on the system as a whole, validating each stage of the concept so that a working system can be implemented, instead of refining each stage individually.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2 provides the relevant literature review on multimedia content descriptions and human motion. Chapter 2 also discusses how motion capture data is reconstructed.

Chapter 3 presents a new query format in the form of Multimedia Query Format (MQF) that was designed to allow multimedia queries using standard description formats such as MPEG-7 and Dublin Core, and to allow multimedia queries using descriptions of human motion. Chapter 3 also detailed the contributions of MQF to the standardization effort of the MPEG-7 Query Format (MP7QF).

Chapter 4 presents a new XML-based description of human motion in the form of Human Motion Markup Language (HMML) that was designed to allow semantic description of human motion in a form that can be queried efficiently.

Chapter 5 presents a method to automatically extract HMML descriptions from motion capture data and presents a result of an investigation into motion similarities between people of different height. Chapter 5 also discusses the possibility of searching human motion using Echoprint [9], an audio search algorithm.

Chapter 6 concludes the thesis and discusses possible future works of the works presented in the thesis.

1.4 Contributions

The main contributions of this thesis are as follows:
1. Introduced Multimedia Query Format (MQF): a new multimedia query format designed to perform multimedia query using multiple multimedia description schemes, i.e. MPEG-7 and/or Dublin Core (Chapter 3).

2. Contributed to the MPEG-7 Query Format (MP7QF) standardization effort (Chapter 3).

3. Presented a detailed analysis on the performance of MQF for querying multimedia databases, which was submitted to MPEG as part of the conformance and performance testing of MQF during the MP7QF standardization process (Chapter 3).

4. Introduced Human Motion Markup Language (HMML): a new, searchable human motion description format based on XML (Chapter 4).

5. Presented motion capture partial reconstruction: a technique to allow feature extraction from motion capture data to allow comparisons of motions (Chapter 5).

6. Presented analysis of walking and running motions to determine the similarities of the walking and running performed by people of different height and walking patterns (Chapter 5).

7. Provided an automatic method to extract human motion descriptions (in the form of HMML and feature vectors) from motion capture data (Chapter 5).

8. Presented an analysis of the effect of the Motion Detection Threshold (MDT) values to the extracted motion feature vector and the implications of varying the MDT values to motion search (Chapter 5).

9. Presented results for motion search using Echoprint [9], an audio search algorithm (Chapter 5).

1.5 Publications

1.5.1 Conference Publications


- K. Adistambha, S. J. Davis, C. H. Ritz, and I. S. Burnett, “Query Streaming For Multimedia Query By Content From Mobile Device,” in International
CHAPTER 1. INTRODUCTION


1.5.2 Journal Publication


1.5.3 MPEG Input Document

Chapter 2

Multimedia Metadata and Human Motion

2.1 Introduction

2.1.1 Multimedia and Metadata

multimedia: “using, involving, or encompassing several media”
media: “a medium of cultivation, conveyance, or expression”
metadata: “data that provides information about other data”
– From the Merriam-Webster Dictionary and Thesaurus [10]

In computing terms, a “media” is an audio or a video recording of an event, while multimedia means that different modality of an event was recorded, which also includes textual description of the event, the audiovisual aspect of the event, and any assorted information presented in any form concerning the event in question. For example, a web page containing an audiovisual recording along with the web page containing information about that event (i.e. Youtube [11]) can be considered multimedia.

The term “multimedia” (in the not too distant past) used to signify that a recording about some event was computerized, and used to be the buzzword of the technology world. Not too long ago, anything described as “multimedia” brings to mind high-tech innovation and progress.

However, the ubiquity of camera phones, fast broadband networks, and high penetration of computers in the consumer market allows not only production houses with big budget to be able to produce multimedia. As evident by the rise of Youtube [4] and similar video sharing sites such as DailyMotion [3] and Vimeo [2] among many, nowadays everyone can record a video on their phones and upload it to the internet for the world to see. Multimedia is now commonplace in what used to be a luxury.
Suffice to say that the rapid pace of media creation leads to new problems. Today we are faced with the staggering statistics of over 72 hours of video uploaded per minute and 800 million viewers per month on Youtube [4]. Similarly, Dailymotion served 110 million viewers per month [12] and Vimeo served 70 million viewers per month [13]. It is not feasible for a single person to consume all of the media produced today. Therefore, a search and description scheme for multimedia data is needed that can answer the questions of:

- what to search?
- how can multimedia be searched on one or multiple servers simultaneously? Especially if the servers in question describes their content using different standards.
- how is the media described in terms of search?
- how can an incoming query be matched to existing data in the database?

This chapter will describe and discuss relevant technologies, historical and modern, in the area of search that could answer the questions posed above. Section 2.2 will describe existing multimedia description formats and how they were used, and Section 2.3 will describe and discuss the search aspect of multimedia.

### 2.1.2 Human Motion

*If we are very observant, we go beyond thinking “How clumsy!” or “How graceful!” and begin to appreciate the appropriateness of the movement invention underlying the idea of the dance.*

– Rudolf Laban, 1974 [14]

Human motion is a subject that goes beyond mere scientific pursuits, but has been associated with the human identity throughout the ages [14]. Dance is a universal language that transcends boundaries, and virtually all human civilization from the ancient times have dances (either dances for aesthetic purposes to express something greater than he is [14] – of which there are many, or war dances designed to scare the opposition à la New Zealand’s Maori people Haka – very popular in rugby [15]).

Suitably to human nature, these “motions” have transcended the computer age, information age, and all upcoming ages and will continue to fascinate us with their intricacies (e.g. dance), their power and accuracy (e.g. sport), and their hilarity (e.g. clowns). Recording and describing them is therefore important and desirable due to our human nature [14].

Empirical observation suggests that the videos uploaded to video sharing sites contains people in the format of instructional videos, video blogs, dancing, singing,
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

etc. To describe these videos, popular video sharing sites employ a relatively basic description format using textual description of the video along with “tags” that summarizes the content of the video. If the video contains human motion, then the motion is described using a textual description. A major disadvantage of textual description is that they are subjective, and cannot adequately describe a complex motion in a standard manner.

Although detailed multimedia description schemes exists (such as MPEG-7 [16], Dublin Core [7], etc), they are lacking a description of human motion. There is no standard that currently exists that can describe human motion (although notation-based standard existed such as Laban’s [14]), especially one that was designed from a computerized search point of view. This is quite a large gap in description standards that begs to be filled.

If music (which is another human identity that spans ages and cultures) can have their notation format to allow music to be written, then why can’t motion? The answer lies in the intricacies of the human body and the complexity of movements that it can perform and the lack of a standard movement terminology [14]. Music has one dimension (time). Motion has four (three orthogonal directions and time). The dance community has tackled this problem of written notation, but in the information age with its mountains of media, these notations need to be updated so that one can search for specific motions.

To create a description for human motion for search purposes, an understanding of how the motion is perceived and described historically is presented in this Chapter. Section 2.4 will discuss how the human body motion is described, and Section 2.4.7 will describe how the human body is represented electronically in the form of motion capture.

2.2 Describing Multimedia

Media (and multimedia) implies that the data is stored in binary format readable only to machines. In order to know what is recorded in the media, one has to open and view it. This is not a problem if the number of media instances is relatively small, but for any reasonably large media database, it is not practical, and even bordering on the impossible for Youtube-like magnitude.

This section describes three methods of annotating multimedia: binary-based (ID3), RDF-based (Dublin Core), and XML-based (MPEG-7). Although there exist other multimedia description schemes besides the three described in this section (seven are described by Smith and Schirling in [17], where ID3 was not among them), the concepts of interest for this thesis in terms of searching multimedia are:

- de-facto standard without a governing body (i.e. a popularity-based standard
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

of which ID3 is a representative),

- general description of content (of which Dublin Core is a representative), and
- detailed description of content (of which MPEG-7 is a representative).

Particularly, any new multimedia search and automated content description extraction method would need to be able to cater for these three general classes of multimedia description.

2.2.1 Binary-based Descriptions: ID3

Since the term “multimedia” is generally understood as computer-based storage of an audiovisual recording (either a picture, a sound recording, a movie, etc.), the multimedia recording is naturally stored as binary data that can be decoded/played back/interpreted by a computer. Therefore, early description of a multimedia data naturally follows the format that the media is stored as (i.e. binary). An example of this type of binary description is embodied in the ID3 format [18] to describe MP3 [19].

Although a binary description is relatively easy for a computer to interpret, it is not so easy for a human reader to interpret without the use of a software to decode the binary data back into a format readable by a human. However, an advantage of a binary-based description is that it is relatively easy to implement from a software perspective.

A significant problem of using a binary-based description is the fact that generally only an English-style ASCII-based alphabet is supported, where one character is represented by a seven bit binary number [20]. Consequently, any description that uses characters beyond what is known in English alphabet requires a workaround, or is impossible to do. For example, ID3 Version 1/1.1 [21] cannot describe media in languages other than English or any languages that do not use the English alphabet, since the structure of the description is rigidly specified in binary terms.

ID3 [18] is a standard primarily used for describing (“tagging”) MP3 [19] files with non-audio information such as the song title, artist name, the release year, the song genre, comments, etc. There is no official international organization concerning the creation and implementation of the standard involving ID3 of any version (unlike Dublin Core and MPEG-7). It is a de-facto standard, created by informal collaboration [21].

In ID3 Version 1, the tags are represented using a fixed 128 characters of information appended to the end of the MP3 file [21]. The 128 characters are separated into fields of specific information about the audio. Although the scheme is relatively simple, usable, and popular at the time, there are two weaknesses of ID3 Version 1:
the fixed 128 characters length, and the fact that the tags are located at the end of the file. As network bandwidth increases and MP3 is starting to become a suitable format for streaming audio, the position of the ID3 tag at the end of the files are becoming a problem, since the information about the song would be received after the song has finished being transmitted.

ID3 Version 2 [22] is a departure from the fixed length Version 1 format (which imposed a strict limit on descriptions which are defined in the previous Version 1 standard). Instead of rigidly specifying the metadata content, the tags are organized into “frames”. For example, a frame describing the information about the song (title, artist, etc.), a frame containing the lyrics, or a frame containing a picture of the album art. The frames are grouped into an ID3 Version 2 container format, and is illustrated in Figure 2.2.1. The maximum size of each frame is 16 MB, and the size of the whole tag is limited to 256 MB. The complete list of officially defined frames is shown in [22].

To solve the streaming problem, ID3 Version 2 prepended the tags at the beginning of the audio content, so that the information about the song would be received before the song is played. Additionally, ID3 Version 2 also has an “unsynchronization scheme” to prevent non-ID3 Version 2 compatible players to attempt to decode the tags as audio [18].

**ID3: Summary**

For such an informal standard, ID3 is highly successful and widely adopted. Virtually all music player applications today support the ID3 standard. The relative simplicity of the description format, coupled with a novel media format that is becoming popular where no metadata information was standardized, drove the rapid adoption and popularity of ID3 even without a governing body behind it.
The success of ID3 highlights that there is strong demand for multimedia metadata, and people are willing to collaborate to create one if it doesn’t exist.

2.2.2 Structured Textual Descriptions

Attempts to force users to deal with information in the same way computers deal with information are doomed to failure.
– Tim Berners-Lee, 1996 [23]

Although binary-format descriptions are relatively simple to implement and parse from a machine point of view due to their use of computer-centric format (as seen in ID3 discussed in Section 2.2.1), it is not quite straightforward for a human to read and modify the descriptions. One method generally employed by binary-based descriptions is to utilize a specialized program to transform the binaries into a human-readable form. Structured textual description formats attempts to solve this human readability problem by using a simple text-based format that is structured in a standardized manner for description purposes.

The most visible example of a structured textual descriptions is the Hypertext Markup Language (HTML) [24] which forms the basis of web pages in the World Wide Web. Following the success of the World Wide Web, a more generalized approach to this method for describing data was created, notably the Resource Description Framework (RDF) [25] and Extensible Markup Language (XML) [26]. Of these technologies, XML can be seen as a generalization of HTML that is able to describe not only content, but general metadata.

Hypertext Markup Language (HTML)

One of the earliest example of a text-based structured description technique is the Hypertext Markup Language (HTML) that was created by Tim Berners-Lee [23], where the core idea of HTML is to separate content and presentation of a document (where both content and presentation cues are written in the same HTML file). HTML formed one part of a larger infrastructure envisioned by Berners-Lee that unifies disparate networks using different information exchange standards that allows the networks to talk using a common language (i.e. the World Wide Web) [24]. Other parts of the Web are the addressing and protocol that enables the Web to unify different computer systems (e.g. PC, Macintosh, X Windows Workstations, etc.) and different servers (e.g. HTTP servers, File servers, News servers, etc.) into one coherent network that speaks a common language [23].

Hypertext Markup Language (HTML) is largely based on the syntax of the Standard Generalized Markup Language (SGML) [27], which was an earlier attempt
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

Figure 2.2.2: John’s online store in HTML format. <h1> denotes a heading, <p> denotes paragraph, and <b> denotes boldface font.

at a standardized document publication format that separates the content and the formatting of the document. HTML was created by Tim Berners-Lee as a way to present content that is cross-platform by describing (in the HTML file) how a piece of text should be formatted and presented to the reader [23].

Bosak and Bray [28] observed that the explosion of content that is published on the web is too much for HTML to handle; that is, HTML only allows one to specify the formatting of a document, and not the semantics of a document. For example John, an online merchant, can have a price list of his goods in his online store. However, using HTML, he is restricted to specify how the price should be displayed on his web page, and no way to actually mark the numbers as “price” (i.e. in HTML, only the words in the documents are taken into account [23]). If he is confident that his price is lower than his competitor’s, he has no way to advertise this lower price (other than writing another HTML page advertising this fact), and there is no standard way for search engines to compare prices (other than having an advanced algorithm that has the necessary heuristics to find and conclude that the wording “price” is comparable across websites).

An example of how John’s store could look like in HTML formatting is shown in Figure 2.2.2, where the tags involved are <html>, <p>, and <b>. A HTML file is denoted by the opening tag <html> and the closing tag </html> and any content between the two opening and closing tags are HTML-formatted content. The content between the tags <p> and </p> specifies a paragraph, and the tags <b> and </b> will print the content between the tags in boldface font. A typical web browser rendering of the HTML code in Figure 2.2.2 is shown in Figure 2.2.3. To a web browser, these information contains only a list of numbers without the context that the numbers relate to price information.

Extensible Markup Language (XML)

Extensible Markup Language (XML) [26] is maintained by the World Wide Web Consortium (W3C) as a text-based information exchange format that is simultaneously human and machine readable. Using a similar syntax to HTML, the emphasis of XML is a structured representation of data that is not limited to how the data
should be presented, but what the data means.

Using the example of John’s Online Store shown in HTML form in Figures 2.2.2 and 2.2.3, an XML representation of John’s store can be rendered in XML using semantic tags to signify the meaning of each element in the data. An example of such a semantic representation is shown in Figure 2.2.4. Structurally, the HTML representation in Figure 2.2.2 and the XML version in Figure 2.2.4 follows a similar pattern: the name of the store, the item, and the item’s price. However, a key difference is the HTML version shows how to format the data for presentation/reading, while the XML version shows the significance of the information contained. Taking the example in Figure 2.2.4, it is quite self-evident that the data is about a store and a list of items with their prices (from the element names <name>, <item>, and <price>). To prepare the data for a web page, one can translate the elements from Figure 2.2.4 into Figure 2.2.2 by defining the mappings of:

- <store> to <html>
- <name> to <h1>
- <item> to <p>
- <price> to <b>

and after the translation, one will arrive at a document almost identical to the HTML version in Figure 2.2.2. This translation process can be assisted using the Extensible Stylesheet Language Transformations (XSLT) [29] standard, which is a standard to perform XML transformations maintained by the W3C.

XML’s flexibility and focus on interoperability in turn allows it to serve as the basis of further standards, which includes:
• Web Service’s Simple Object Access Protocol (SOAP) [30] and Web Services Description Language (WSDL) [31]

Web Service is a standard to allow data interchange between computer systems that is maintained by the W3C [32]. SOAP is the data interchange method, and WSDL is the service specification published by a network-facing computer to allow other computers to utilize the services provided by said computer. Both standards uses XML as their data interchange language.

• Really Simple Syndication (RSS) [33] and Atom [34]

RSS [33] and Atom [34] are methods to publish articles on the web that can be syndicated (i.e. aggregated, captured, or processed) by other sites. RSS is maintained by the W3C [35], and Atom is maintained by the Internet Engineering Task Force (IETF) [36].

• Extensible Hypertext Markup Language (XHTML) [37]

XHTML is a redefinition of HTML using XML conventions, and is maintained by the W3C [37]. Since XML requires well-formed documents for a valid document (e.g. any opening tag should have a matching closing tag), XHTML forces HTML to be as well-formed as XML (where previously in HTML, an opening tag does not require a closing tag to be parseable by a browser).

• Extensible Messaging and Presence Protocol (XMPP)[38]

XMPP is a standard for real-time messaging in the Internet that is maintained by the IETF [38]. Notable users of this standard includes Google Talk [39] and Facebook Chat [40].

• XML Schema [41]

XML Schema [41] describes the rules for how an XML document should be structured to facilitate interoperability. For example, an XML-based address book could contain the element `<name>`, while other implementation could separate the `<firstName>` and `<lastName>`. To solve this interoperability
problem, a schema could be defined that contains an element <name>, which in turn has child elements <firstName> and <lastName> to unify the two address book descriptions.

- Extensible Stylesheet Language Transformations (XSLT) [29]

XSLT is a standard maintained by the W3C to perform transformations of XML document from one form to another by defining a set of rules for each element of the input XML document and how the output XML document should look like [29]. The XSLT transformation rules are written in XML format; Figure 2.2.5 shows an example of an XSLT transformation rules required to transform Figure 2.2.4 (an XML document) into Figure 2.2.2 (a HTML document).

The example XSLT shown in Figure 2.2.5 instructs an XSLT parser to transform the incoming XML document using the rules of:

- any <name> element contains <item> elements, which is to be transformed using a rule defined in a <template> element defined elsewhere in the document.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">

    <xsl:output method="xml" indent="yes" encoding="UTF-8" />

    <xsl:template match="/store">
        <html>
            <h1><xsl:value-of select="name" /></h1>
            <xsl:apply-templates select="item" />
        </html>
    </xsl:template>

    <xsl:template match="item">
        <p>
            <xsl:value-of select="." />
            <b><xsl:value-of select="../price" /></b>
        </p>
    </xsl:template>

</xsl:stylesheet>
```

Figure 2.2.5: XSLT code used to transform the XML of Figure 2.2.4 into Figure 2.2.2.
– in the `<template>` element (which contains the instructions for transforming any `<item>` element), any `<price>` element should be encapsulated inside a `<b>` element (which would result in any price information to be rendered in boldface font).

XML Fragment Request Unit (FRU) and Fragment Update Unit (FUU)

Processing XML documents often requires the XML document as a whole to be present locally. If a client would like to process a small part of an XML document located elsewhere, the document as a whole would have to be transmitted before any processing could take place. For example, if an XML document contains 1000 nodes and the client is only interested in processing the information contained in only one node. The other 999 node would be transmitted and discarded, resulting in processing inefficiencies.

To overcome this inefficiency in XML processing, MPEG-7 standardized a method for the delivery of XML fragments of MPEG-7 descriptors known as The MPEG-7 Systems Method for Textual Encoding (TeM [16]). TeM offers a method for delivering XML fragments contained within Fragment Update Units (FUUs), where these FUUs instruct the receiver where to insert/update the fragment of XML or delete the node (and thus all associated descendant nodes, due to the hierarchical nature of XML).

To complement FUU and to provide the capability of requesting parts (i.e. frag-
ments) of a remote XML document, Fragment Request Units (FRUs) [42] were thus standardized. FRU provides the user with a technique to request only the relevant fragments of an XML document to be transmitted without the need to transmit the entire XML document. The user or application decides which fragments are to be retrieved either through use of a known information (i.e., an XML Schema) or from information based on already received FUUs. FRUs can request fragments from a remote XML document by FRU navigation or query. FRU navigation allows stepping through the XML structure either on a node-by-node basis (i.e., one node at a time) or on a level-by-level basis (i.e., all immediate child nodes of a selected node), retrieving only the relevant nodes or levels. FRU query allows a query (i.e., an XPath expression) to be sent and performed on the remote XML document where only the results of the query are retrieved.

The results of FRU operations are delivered back to the originator through FUUs. FUUs contain context path information necessary for the user to reassemble the XML fragments while preserving the original XML structure. A block diagram illustrating the operation of FRUs and FUUs is shown in Figure 2.2.6.

In the diagram in Figure 2.2.6, both the client and the server have a separate XML document (with a larger, more complete XML document residing in the server). The XML document in the server contains two nodes: in this case, nodes X1 and X2. The client would like to pull only the node marked X2 in Figure 2.2.6 without having to receive a copy of the server’s document as a whole due to bandwidth constraints. To achieve this, the client constructs the FRU (that contains the path specification for the node X2) which is transmitted to the server. The server receives and decodes the FRU, and according to the path specification in the FRU, navigates to the X2 node and sends only the contents of the requested node to the client in the form of a FUU.

For example, the server has a large MPEG-7 document (named “MPEG7Description.xml”), and the client would like to extract the contents of the node “//Classification” within. First the client constructs the FRU detailing the name of the XML document and the desired path (shown in Figure 2.2.7). The server processes the request, navigates to the relevant section of its MPEG7Description.xml document, and transmits the contents of that node to the client (shown in Figure 2.2.8).

Therefore, FRU and FUU work in combination to allow a client to directly request parts of an XML document that resides in the server, hence allowing processing to take place without the need to transmit the full XML document.

**Binary representation of XML documents (BiM)**

A drawback of using XML is the textual nature of XML which is relatively verbose compared to binary representation of data. While this textual representation of
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

Figure 2.2.7: An example Fragment Request Unit (FRU) requesting a part of a document called “MPEG7Description.xml”. The FRU is requesting the node called “Classification”.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FRU>
  <Src>MPEG7Description.xml</Src>
  <Query>//Classification</Query>
</FRU>
```

Figure 2.2.8: An example reply from the server using Fragment Update Unit (FUU). The FUU describes that the requested “Classification” node exists in the path of “/Mpeg7/Description/CreationInformation/Classification” and returns the contents of the node.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FragmentUpdateUnit>
  <FUCommand>addNode</FUCommand>
  <FUContext>
    /Mpeg7/Description/CreationInformation/Classification
  </FUContext>
  <FUPayload>
    <Descriptor>
      <Statement mimeType="text/plain">
        <CaptionLanguage>en-US</CaptionLanguage>
        <CaptionLanguage>fr</CaptionLanguage>
        <CaptionLanguage>de</CaptionLanguage>
      </Statement>
    </Descriptor>
  </FUPayload>
</FragmentUpdateUnit>
```
data provides XML with human-readability, platform-independence and language-independence, it is bandwidth-consuming to transmit XML in its entirety in its native textual format. As a solution to this, MPEG provides BiM (Binary MPEG format for XML), which is part of the MPEG-B standard ISO/IEC 23001-1 [43] for binary representation of XML data.

Although many compression schemes are available that perform efficiently with textual data (such as the popular Lempel-Ziv algorithm [44]), the drawback of such general-purpose compression schemes is the lack of awareness of those algorithms to the structure of XML. Such general-purpose algorithms accept a block of (textual) data as the input, and produces the compressed version as the output. It is not possible to extract a specific node from the compressed version of the document without uncompressing the whole document first. In contrast, specialized XML compression tools such as BiM exploits the hierarchical nature of XML, and thus able to preserve the structure of the document and identify a specific node, even in compressed form. For example, a node named `<ContentInformation>` could be represented using a single binary digit if the schema requires the presence of that node.

Resource Description Framework (RDF)

Resource Description Framework (RDF) [25] is a web-centric standard to describe a resource located on the internet (e.g. the author, title, etc. of a web page) that is maintained by the World Wide Web Consortium (W3C) [45, 25]. However, the concepts of RDF can also be used to describe any other resources that can be described using the Subject-Predicate-Object construct.

The following example is derived from [25]. For a person known as Dr. Eric Miller with an email address of em@w3.org, some logical connections can be made, such as:

- The person’s name is Eric Miller.
- His title is Dr.
- His email is em@w3.org

These attributes are describing the same person, and hence can be visualized as a graph as shown in Figure 2.2.9. Figure 2.2.9 highlights the definitions of the Subject-Predicate-Object constructs similar to natural languages; the graph defines a contact form, where each box in Figure 2.2.9 defines a Subject matter or an Object, and the connecting arrows defines the Predicate that applies to those Subjects or Objects. For example, the main Subject of the information in Figure 2.2.9 is
Figure 2.2.9: An example of RDF graph describing Dr. Eric Miller, his email address, full name, and title (example taken from [25]).

http://www.w3.org/People/EM/contact#me, meaning the graph is about a contact information, and the type that is denoted by the arrow marked by:

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

is a person that is denoted by the box:

http://www.w3.org/2000/10/swap/pim/contact#Person

The contact person’s name, title, and email address are identified by their respective predicates, and the information contained within those predicates are “Dr.”, “Eric Miller”, and “em@w3.org”, respectively.

Uniform Resource Identifier (URI)

To describe the relationship between entities, RDF utilizes the Uniform Resource Identifier (URI) [25, 24] which is shown in Figure 2.2.9 as strings that begins with “http://”. The relationship between Uniform Resource Identifier, Uniform Resource Name, and Uniform Resource Locator (also known as web address) are:

- Uniform Resource Locator (URL): a unique location identifier of a resource and was the core addressing scheme of the web, where it was postulated in [46] that “global naming leads to global network effects” with the idea that a
globally unique location name would lead to a global network.

Examples:

- ftp://ftp.is.co.za/rfc/rfc1808.txt

URLs are separated by slashes ("/"), where the first part of a URL is the network protocol required to access the resource (e.g. http and ftp in the examples above), the second part is the server name (mpeg.chiariglione.org and ftp.is.co.za in the examples above) and the rest of the string identifies the location of the resource within the server (standards/mpeg-7/mpeg-7.htm and rfc/rfc1808.txt in the examples above).

- Uniform Resource Name (URN): a unique name identifying a resource that is not limited to the Internet (e.g. could be a physical resource).

Examples:

- urn:mpeg:mpeg7:schema:2001
- urn:isbn:0471486787

A URN is identified by the string urn, immediately followed by the namespace of the resource. In the examples above, the URN urn:mpeg:mpeg7:schema:2001 identifies the MPEG namespace for the MPEG-7 standard, and the URN urn:isbn:0471486787 the namespace is the International Standard Book Number (ISBN) with the number 0471486787 identifying a specific book “Introduction to MPEG-7: multimedia content description interface”.

Formally, the URN namespace (i.e. mpeg, isbn) is managed by the Internet Assigned Numbers Authority (IANA) and is listed in [47].

- Uniform Resource Identifier (URI): a superset of both URL and URN to provide a globally unique identifier (either name, location, or both of a resource). A URI is a superset of both URL and URN. The relationship of URI, URL, and URN is illustrated in Figure 2.2.10.

In conclusion, the graph representation of Dr. Eric Miller’s contact information shown in Figure 2.2.9 is instantiated in XML form in Figure 2.2.11.

**Structured Textual Descriptions: Summary**

Flexibility and extensibility are the central theme of structured textual descriptions, which is evident by the use of XML [26] to describe different types of data (presentation e.g. XHTML [37], syndication e.g. RSS [33], communication e.g. Web Services
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

24

Uniform Resource Identifier

Uniform Resource Locator

Uniform Resource Name

http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm

urn:isbn:0471486787

Figure 2.2.10: The relationship between Uniform Resource Identifier (URI), Uniform Resource Location (URL), and Uniform Resource Name (URN). Both URL and URN are subsets of URI.

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>

Figure 2.2.11: XML-based representation of the RDF graph in Figure 2.2.9 [25].
and perform tasks not directly related to metadata or any structured descriptions (e.g. XMPP [38]). XML can even be used to describe itself (e.g. XML Schema [48, 41]) and transform itself (e.g. XSLT [29]).

RDF [25] focus is on logical connection of concepts. It is structured in the Subject-Predicate-Object construct, and can be used to infer additional logical connections between things that are described using RDF. Although not strictly a structured textual description in the sense of XML, it can also be expressed in XML among other formats.

2.2.3 RDF-based Multimedia Descriptions: Dublin Core

Dublin Core [7, 49] is an RDF-centric approach to metadata maintained by the Dublin Core Metadata Initiative (DCMI), where the core metadata descriptions are standardized as ISO 15836:2009, ANSI/NISO Z39.85-2007, and IETF RFC 5013. Although originally designed to describe textual data such as web pages, the generality of the descriptors made Dublin Core also suitable for multimedia description [50]. The key effort of DCMI is defining the usage and semantics of each keyword that forms Dublin Core (shown in Table 2.2.1) to ensure consistent semantics of Dublin Core’s keywords across different applications.

Unlike the binary-based metadata format of ID3, Dublin Core defined the descriptors and the contents to be associated with those descriptors, and refrained from rigidly specifying the techniques of how the descriptors and descriptions should be represented. Consequently, one can describe a media using Dublin Core stored in a multitude of manners, such as RDF, XML, plain text, or key-value, among many possibilities.

The current version of Dublin Core (version 1.1) [7] as published by the Dublin Core Metadata Initiative (DCMI) defined fifteen descriptors shown in Table 2.2.1.

Although Dublin Core can be expressed using many different schemes (as long as the semantics of the descriptions remains within the boundaries defined by Dublin Core), nevertheless DCMI defined some recommendations of using Dublin Core with the most popular formats:

- Text-based descriptions using DC-TEXT recommendation (recommended in [51]).
- HTML/web-page descriptions using HTML recommendation (recommended in [52]).
- XML-based descriptions recommendation (recommended in [53]).
- RDF-based descriptions recommendation (recommended in [54]).
<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contributor</td>
<td>An entity responsible for making contributions to the resource.</td>
</tr>
<tr>
<td>coverage</td>
<td>The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant.</td>
</tr>
<tr>
<td>creator</td>
<td>An entity primarily responsible for making the resource.</td>
</tr>
<tr>
<td>date</td>
<td>A point or period of time associated with an event in the lifecycle of the resource.</td>
</tr>
<tr>
<td>description</td>
<td>An account of the resource.</td>
</tr>
<tr>
<td>format</td>
<td>The file format, physical medium, or dimensions of the resource.</td>
</tr>
<tr>
<td>identifier</td>
<td>An unambiguous reference to the resource within a given context.</td>
</tr>
<tr>
<td>language</td>
<td>A language of the resource.</td>
</tr>
<tr>
<td>publisher</td>
<td>An entity responsible for making the resource available.</td>
</tr>
<tr>
<td>relation</td>
<td>A related resource.</td>
</tr>
<tr>
<td>rights</td>
<td>Information about rights held in and over the resource.</td>
</tr>
<tr>
<td>source</td>
<td>A related resource from which the described resource is derived.</td>
</tr>
<tr>
<td>subject</td>
<td>The topic of the resource.</td>
</tr>
<tr>
<td>title</td>
<td>A name given to the resource.</td>
</tr>
<tr>
<td>type</td>
<td>The nature or genre of the resource.</td>
</tr>
</tbody>
</table>

Table 2.2.1: Dublin Core version 1.1 descriptors (source: [49]).
2.2.4 XML-based Multimedia Descriptions: MPEG-7

Although MPEG-1 [19], MPEG-2 [43], and MPEG-4 [55] standards concern the representation, compression, and transport of media content, MPEG-7 [16] aims to solve the description problem of multimedia. Of particular interest in terms of this Thesis is the 5th part of MPEG-7: Multimedia Description Schemes.

Figure 2.2.12 illustrates the scope of MPEG-7: namely, the MPEG-7 standards standardized the description aspect of multimedia metadata, and do not govern how the descriptions are generated and consumed. This is similar to older MPEG-1 standard [ref] (which contains the popular MP3 [ref] audio compression standards), where in MP3, only the decoder is standardized and not the encoders. This approach allows third-party implementations to create better implementations of the encoders.

Figure 2.2.13 illustrates the elements of MPEG-7 which contains major parts of:

- Description Definition Language (DDL): DDL is the method with which Descriptors are instantiated. To fulfill the needs of MPEG, basic XML Schema
2.3 Searching Multimedia

... searches frequently produce a lot of junk because they generally take into account only the words documents contain and have little or no concept of document usefulness or quality.

– Tim Berners-Lee, 1996 [23]

In 1996, Tim Berners-Lee observed [23] that textual searches without context generally returns, in his own words, “junk”. It is ironic that this statement remains more or less true even in 2012. Although search technologies have advanced by leaps and bounds compared to the state that it was in 1996, search generally remains text-bound.
Multimedia search is especially problematic due to the textual nature of search. There is no simple method to accurately describe what is contained in a multimedia data. Video sharing sites partially solved this problem by using “tags”, which are one-word descriptions of the content of a video being uploaded. The use of tags mitigates, but does not solve, the problem of multimedia search.

There are standardized multimedia descriptions available (Dublin Core [49], MPEG-7 [16] as discussed in Section 2.2), unfortunately, HTTP [58] remains the protocol of choice in search due to the open-ended nature of HTTP that enables it to transmit data without regard to its content [23]. HTTP is a communication protocol that could be used to transmit multimedia queries (i.e. transporting a Dublin Core or MPEG-7 description of a desired multimedia).

2.3.1 Web-based Technologies

At the lowest complexity level of multimedia query transport, standard web-based technologies for information exchange are used, such as the HTTP protocol [58]. In its simplest form, a multimedia query is encoded in HTML Form [24], where the endpoint of the form is an entry point of a multimedia query system. Communications using HTTP can be performed using stateful or stateless (Representational State Transfer - REST) [59] methods, depending on the design of the multimedia database in question. The term “stateful” refers to the requirement for the server to retain the state of a client’s query during a query session (where the state is kept until either the query is finished or timed out) [60]. In contrast, “stateless” means that the query state is not retained in the server, and thus the server treats each query as a new query (which is the original design of HTTP [58]). The advantage of a stateful approach is the ability to query within a previous result set, while the disadvantage is an increase in the server’s resource usage (which depends on the number of clients connected to it at the time). For a stateless approach, the advantage is less resources used for each client due to the lack of need to retain a client’s connection, thus potentially a server of similar capabilities could serve more clients compared to the stateful approach. However, the disadvantage is the increase in communication overhead due to each query is treated as a new query in the server.

The advantage of using standard HTTP-based transport is implementation simplicity in the transport. However, the disadvantages are the server dependence on the queries. Since there are no standard governing multimedia queries for HTTP and the HTTP Form was designed as a basic client-to-server information exchange, each multimedia database can implement their own method of query. This will result in a considerable barrier when more than one database is involved in a query, since the capabilities of each server would have to be taken into account. Furthermore, there
is no standard method to query a server on its multimedia query capabilities.

## 2.3.2 Query-by-example

In a query-by-example, a query is formed by using a media representation of the desired result to serve as the query term. For example, in Figure 2.3.1, the user would like to search for an image, and presents an example of how the image should look like to the server. The server would search its database for an image similar to the one the user presented (using tools such as image similarity measure, color similarity measure, etc.). An example of a query-by-example using image as the query term is [61].

An advantage of using a media as a query term instead of a more traditional text-based queries is the lack of need for the user to describe the desired match. For example (as in Figure 2.3.1), the user could “show” the server what he meant instead of having to enter “picture of a landscape with blue sky and green grass” as a query, due to the subjectivity of textual descriptions (the server could objectively examine the user’s example instead).

## 2.3.3 Fingerprinting

Fingerprinting is a technique to perform query-by-example, where the objective is to retrieve a single match (the original media), and the example given is a distorted version of the media. For example, a user hears a song in a public address system and would like to know the title and artist of the song. He proceeds by recording the song using his mobile phone, and sending the recorded song to an audio fingerprint matching service. The service matches the (distorted) example of the song to its database, and replies to the user with the requested details about the song. Popular applications in audio include [62, 9], and popular technique in image is SIFT [63].

![Image query by example](image.png)

Figure 2.3.1: Image query by example.
Audio fingerprinting system such as Echoprint performs matching in the client side by processing the recorded audio and extracting certain features, and on the server side by comparing the extracted features to its database. The extracted features are thus required to be robust enough to be identifiable even with the addition of noise. The Echoprint audio fingerprint is stored in feature-timestamp form. An example of a fingerprint is shown in Figure 2.3.2.

The codebook contains fingerprints of known audio files, where an example of a codebook is shown in Figure 2.3.3.

Detailed description of the Echoprint algorithm is depicted in Appendix C.

2.3.4 Human Motion Oriented Search and Classification

Video-based Techniques

Most recently, VideoMocap by Wei et al [64] provides a method to extract a motion capture-like 3D representation of the human body from 2D video. This is achieved by manually annotating the joints in keyframes in the video, and by using physics-based interpolation method to reconstruct the 3D skeleton of the human body accordingly.

The work by Gu et al [65] further performed action and gait recognition on reconstructed 3D model of the human body from video using a Hidden Markov Model (HMM) based approach; the reconstruction of the 3D human body by Gu et al combines manual annotation of the body joints with a hierarchical skeleton model similar to motion capture representation of the human body.
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

Motion Capture-based Techniques

Since the advent of motion capture technologies and the relatively low cost of processing power nowadays, people have been capturing an increasing amount of motion for games development and movies, and hence the problem of how to catalogue the motion and to search them is becoming increasingly relevant. The state of the art in this area is the work of Muller et al [66, 67] and Guerra-Filho et al [68, 69]. Muller developed the concept of “motion template”, where a number of motion capture involving a motion are analyzed and their common features recognized. The result of this analysis is the template for that specific motion. Some tolerance was designed in the system, so that minor differences in movements would still be recognized as the same motion to address the problem of the exactness of motion capture (hence the term “template”).

In contrast, Guerra-Filho developed the Human Action Language that takes inspiration from written language and defines a motion as a series of small actions connected together to form larger, more complex motions. This concept is called “Human Action Language”. The key concepts of the Human Action Language are compactness of description (describing an action with the least amount of symbols, called atoms), view-invariance (a 3D motion should be able to be projected into a 2D plane), selectivity (the ability to differentiate between different atoms) and reconstructivity (reconstruction of a described motion back into its motion capture representation).

Barbic et al [70] explores the use of Principal Component Analysis (PCA) and Gaussian Mixture Model (GMM) to perform automatic segmentation of motion capture data, due to their observation that a motion capture session tend to get longer as more motion is captured, especially if natural behavior of the actor during the capture session is desired. Barbic et al achieved high accuracy using PCA approach.

Similarly, Li et al [71] explored the use of Singular Value Decomposition (SVD) and Dynamic Time Warping (DTW) distance measurement to perform automatic segmentation.

2.3.5 Multimedia Search and Description: Summary

From Metadata Description and Transport Point of View

A scalable multimedia query system requires a standardized set of descriptions such as MPEG-7 or Dublin Core. While the requirements for standard descriptions are satisfied, the transport of such descriptions to perform a query on a multimedia database is severely limited today.
Major centralized multimedia databases such as YouTube reverts to the lowest common denominator available currently, that is, HTTP-based Form queries. While this approach works with heavily centralized systems, it is not scalable to multiple, smaller databases due to the lack of standard on what should constitute a multimedia query and how it should be transported.

Efforts to overcome this limitation were performed on many fronts, such as libraries and MPEG. However, certain limitations still exists, in that each solution was applicable to their own separate set of problems. For example, library-based solutions are optimized to work only in library environment, SQL-based solutions are not easily extensible, and MPEG-7 Query Format was designed specifically to query MPEG-described multimedia databases.

Hence, an extensible, scalable, and simple multimedia query transport format that transcends a specific problem space is desired. The existence of such a format would allow not only queries to a highly centralized multimedia database, but enables small personal multimedia databases to be connected and queried as a whole.

From Human-Oriented Search Point of View

Although significant progress was made in the area of human motion from a numerical point of view using GMM, PCA, SVD, HMM, or physics methods, annotating motion data in a structured manner (from either video or motion capture) for search purposes is noticeably absent. Advances in the form of Human Action Language and Motion Templates are bridging the gap between semantic and numeric descriptions of human motion, but both technologies are not designed with manual generation of annotation, instead relying on numerical methods to provide the semantic descriptions of motion capture data.

2.4 The Human Body, Motions, and Descriptions

This Section will review techniques for describing human motion, both using symbolic representations of motion (much like musical notation represents music), and from recording human motion in three-dimensions (motion capture). This Section also provides information for reconstructing a motion capture recording by using the popular BVH motion capture format as an example.

2.4.1 Human Perception: Point Light Analysis

Gunnar Johansson experimented with Point Light animation as early as 1973 [72] by using retroreflective material fixed to the joints of an actor, using a total of ten points that was fixed to the neck (one point), the elbows (two points), the wrists (two
Figure 2.4.1: Illustration of a Point Light Analysis of human movement. In this example, the points are located in the major features of the body: the head, the shoulders, the elbows, the wrist, the chest, the hips, the knees, and the foot.
points), the hip (one point), the knees (two points), and the ankles (two points). The actor was instructed to walk parallel to the camera, so that the motion was of a person walking as seen from the side (similar to Figure 2.4.1). The motion was captured using a video camera and a floodlight that was placed adjacent to the camera (where the retroreflective material will show up as bright spots in the recorded video), and was played back by manipulating the contrast of the images so that only the dots will be visible to an observer. Johansson observed that without fail, observers can immediately recognize a human walking motion given only one second of video. However, when the movement was stopped, the observers cannot recognize the dots as representing a human. Further experiments was immediately performed by Johansson by instructing the actor to walk toward and away from the camera with the same result of the observer recognizing the moving dots as human walking. Also, Johansson further proved that the Point Lights in motion can be recognized easily by observers by instructing the actor to perform motions other than walking (e.g. running, cycling, climbing, dancing, gymnastic motions, etc.) where the motions performed were accurately identified by the observer [72]. Note that this is one of the earliest study using Point Light movements that was performed in the literature, therefore the observers would have no previous experience of seeing the dots in motion.

For a live example of the Point Light animation in motion, BioMotionLab [73] provides an animation that can be modified to represent different gender, weight, and mood of the Point Lights of walking motion.

Another interesting experiment using Point Lights was performed by Shim et al. [74] that shows that a person can estimate the weight of a box being lifted by a person shown using only point light display by judging the effort they expended. This result is especially accurate if the size of the subject lifting the weight is known (since Point Light display gave no reference to the size of the person being recorded). Also, Cutting [75] showed that human perception can still recognize a Point Light motion even when the points are masked by noise, making the motion recognition relatively robust to disturbances.

### 2.4.2 Anatomical Description of the Body

From an anatomical viewpoint, there are three planes dividing the human body, where each plane divides the body into two halves (Figure 2.4.2):

1. the sagittal plane (also known as the wheel plane) divides the left and right halves (lateral portions) of the body.

2. the coronal plane (also known as the door plane) divides the front part (ventral) and back part (dorsal) of the body.
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

Figure 2.4.2: The planes dividing the human body: The Sagittal plane describes movement in forward-backward directions, the Coronal plane describes movement in left-right directions, and the Transverse plane describes up-down movements (Graphics by Mrabet [76] Licensed under Creative Commons Attribution and Share-Alike license (CC-BY-SA)).

3. the transverse plane (also known as the table plane) divides the upper (cranial) and lower (caudal) parts of the body.

Due to the complexity of human motion, dividing the body into the sagittal, coronal, and transverse plane and describing the movements of the body in terms of those planes is the most natural method.

In three-dimensional terms, the movement of the human body can be described in terms of the three planes. For example, from a standing position with the arms hanging at the side, moving the arms to reach forward consists of movement in the sagittal plane toward the ventral direction, and movement in the transverse plane to the cranial direction. Similarly, moving the arms to reach to the side consists of movement in the coronal plane laterally and movement in the transverse plane cranially. In other words, the three planes (sagittal, coronal, and transverse) provide the Cartesian planes of the human body.
In terms of the anatomical body planes, the direction of movements of the body are classified as such [77]:

- right and left describe movements toward the right and left in the coronal plane.

- anterior and posterior are movements toward the front (anterior) and the back (posterior) in the sagittal plane.

- superior and inferior are movements upward (superior) and downward (inferior) in the transverse plane.

For the leg movements, Figure 2.4.3 shows possible movements of the right leg in the three planes described in Figure 2.4.2:

- abduction and adduction are the left and right movements in the coronal plane. Abduction is a movement away from the center of the body, and adduction is a movement toward the center of the body.
• external and internal rotations are movements in the transverse plane. External rotation is a movement away from the center of the body, and internal rotation is a movement toward the center of the body.

• flexion and extension are movements in the sagittal plane. Flexion is a movement where the joint angle decreases (flexes), and extension is a movement where the joint angle increases (extends).

2.4.3 Text-based Descriptions

The naive method of describing human motion is to describe the motion in text. For example:

"The person takes four steps forward beginning with the right leg, and takes four steps backward starting with the left leg. The hands are moving in sync with the leg movements. The motion starts and ends with the body standing straight."

Although the description above is sufficiently detailed to describe a motion, the description is relatively verbose, and the smallest details about the motion would have to be recorded to describe an exact motion [14]. For example, the understanding of a "step" can be different from person to person, where using the description above, a "step" can be defined as "a step forward of normal length for the person" [14]. For a sufficiently complex motion with rigid specifications of how the motion should be performed (such as in dance), the textual descriptions can be very verbose and hard to read at a glance. Furthermore, the sentence "the hands are moving in sync with the legs" can potentially create confusion, How in sync? Naturally in sync (e.g. the left arm moving in the opposite direction of the left leg?) or movement-wise sync (e.g. the left arm moving in the same direction of the left leg?). Also, if the four steps performed are not uniform, but instead alternates between short and long steps, the above description must be rewritten as:

"The person takes four steps starting with the right leg, where the steps are alternating between small steps and large steps, and takes four steps backward of normal length using a similar timing to the normal forward steps. The hands are moving in sync to the same direction of the same side’s legs. The motion starts and ends with the body standing straight."

Immediately, the description becomes verbose, confusing, and the more precise the description is, the longer the description that is necessary. Further, the description above is only valid of written by one person. Another person describing the same motion might not necessarily made the same choice of words.
Rudolf Laban [14] describes another version of the example above, along with the difficulty to describe any movement in textual manner, especially in the field of dance.

Unfortunately, in the context of Youtube and other popular video sharing sites available today, these textual descriptions are all that is available to the user. Any motion that is more precise than “walking”, “running”, “jumping”, etc. cannot currently be described.

2.4.4 Notation-based Descriptions

As a response to the difficulty in describing motions using words (as demonstrated in Section 2.4.3) and the lack of standard regarding the notation for describing human movement coupled with a desire to have a motion notation similar to musical notation, dance notations to provide a written notation for motion were developed [14]. Recording and understanding human motion in written form was the underlying theme in the development of dance notations, such as the Labanotation system [78].

The most popular dance notations are Laban and Benesh, and there exists other, comparatively less popular notations such as such as Eshkol-Wachman and Beauchamp-Feuillet. In particular, Labanotation received popular attention from researchers in the computing field (partial list of motion technologies using Labanotation as a basis are [79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89]). Labanotation [14] was developed in 1920 by Rudolf Laban to record dance movements using a series of symbols written on a special staff where each place in the staff denotes a major limb such as both legs, both arms, the torso, and the head. The notation is read from bottom to top. The reason for using symbols was to provide a language-free dance notation using relatively simple symbols due to Laban’s observation that trying to describe a motion using textual description was error-prone and highly inefficient. Laban’s notation only writes the movement, i.e. a body kept still in space is not written. Furthermore, Laban developed a series of concepts based on “effort”, that is, the intended meaning of a motion. For example, reaching out to touch someone and to punch someone involve relatively the same movement of the hands, however the effort and the goal of both motions are very different. Note that Labanotation describes movements using the three anatomical planes previously described in Section 2.4.2 (sagittal, coronal, and transverse). The sagittal and coronal movements are written together as the nine directional symbols, while the transverse plane is encoded in Labanotation using the three levels (low, medium, and high).

Contrary to Laban’s movement approach, Benesh movement notation uses standard five-line musical staff to describe a series of poses, and is mainly used to notate
ballet. Since both Laban and Benesh methods were designed for dance, their notation symbols are closely tied to written music concept of beats and measures. Searching using direct application of dance notations is problematic, since the techniques are designed to be human-centric in their approach in writing and reading.

Labanotation specifically skips over motions that are considered “natural” in order to improve human readability (e.g. the most natural way for the extended hands to move from the side of the body to the front follows a circle and not a straight line). An example of Labanotation is shown in Figure 2.4.5. Note that in the example in Figure 2.4.5, walking forward, walking backward, and running movements are written only as leg movement. It is assumed in Labanotation that the rest of the body will naturally follow the leg movements in all three cases (where the arms swing naturally according to the leg movements) and thus does not need to be written. What is written is the most important essence of the movements (in this case, the legs).

LabanXML [86] is an XML representation of the Labanotation using Document Type Definition (DTD) [90]. MovementXML [84] is a more modern representation of Labanotation in XML using XML Schema. It extends LabanXML by providing a more thorough XML interpretation of Labanotation as described in [78]. Further, by using XML Schema, the constructs of MovementXML is relatively more sophisticated compared to LabanXML. However, by directly translating Labanotation into XML, MovementXML suffers from the same drawback as Labanotation for describing computer-based multimedia content; namely, the lack of a method to describe a motion in a rigid timescale (e.g. frames per second), and the lack of a method to search for a certain movement. Labanotation, and thus MovementXML, was created for human consumption and was designed with readability of a motion description in mind, not automation and search which are the primary goals of this thesis.

2.4.5 Motion Capture

Motion capture is a method of recording human motion in a very detailed manner involving all three dimensions, where the position of each sensor in the motion capture system are sampled at hundreds of times per second (generally between 100 Hz – 120 Hz). Figure 2.4.6 is an illustration of the motion capture process, where the figure is similar to the point light display shown in Figure 2.4.1, with the addition of the “lines” connecting the dots. These lines are arranged in a hierarchical manner to represent the bones in the human skeleton; a key difference in a motion capture system and point light display is that point light display relies on human interpretation of the movements of the dots (and thus was used to perform researches on human perception of motion), while motion capture explicitly records what each
Figure 2.4.4: Labanotation symbols and staff [78]. The dotted line in the staff is not drawn. The notation is read from bottom to top.
Figure 2.4.5: Example of Labanotation for the motion scenario in Section 2.4.3: “The person takes four steps starting with the right leg, where the steps are alternating between small steps and large steps, and takes four steps backward of normal length using a similar timing to the normal forward steps. The hands are moving in sync to the same direction of the same side’s legs. The motion starts and ends with the body standing straight.” Labanotation is read from bottom to top; where part (b) is a continuation of part (a).
Figure 2.4.6: Illustration of motion capture of a walking motion. Although similar to point light analysis of human movement (shown in Figure 2.4.1), motion capture utilizes the knowledge of the skeletal structure of the body. The point lights are connected by lines that show the bones.
dot represents and arrange them into a hierarchy that is based on human skeleton.

Until recently, performing motion capture was prohibitive in terms of time and investment required as the most common method was to use a body fitting costume (motion capture suit) with markers with the motion recorded using multiple cameras, such as in the highly accurate Vicon motion capture system. Another type of suit made by Moven Technologies [91] utilizes wireless sensors in place of the optical sensors which essentially removes the requirement of the Vicon [92] system by allowing a capture session not to depend on camera positions and thus potentially allows a larger capturing area.

Similar to motion capture, the MPEG-4 Face and Body Animation (FBA) standard [55] defines an overarching human body animation standard that includes face animation as well as body animation. Advantages of MPEG-4 FBA compared to motion capture formats in terms of human body representation include a standardized skeleton hierarchy, texture mapping, and scalable animation detail. One of the focuses of the standard is faithful representation of the human motion using a standardized body model instead of letting the user define their own skeleton hierarchy, which solves the problem of having to redefine the skeleton for different applications (as in most popular motion capture formats) that could lead to interoperability problems when two similar motions are recorded using slightly different skeleton hierarchy.

Recently, Microsoft released the Kinect [93] which is a cheap and simple method for motion capture in a relatively small package. Although Kinect was originally developed for gaming applications, it is now being used for motion capture with a relatively low barrier of entry [10] as there is no requirement for the user to wear a dedicated motion capture suit, which are typically time-consuming to set up and expensive.

The problems of using motion capture data to perform human motion search are due to the fact that there is no standard on the human skeleton hierarchy other than MPEG-4 FBA (although typically they are quite similar, i.e. some system records the position of the thumb, and some system does not) and the difficulty in reconstructing a motion capture values from a video.

2.4.6 Kinematics and Human Motion

Kinematics is a study of movements without regard to the forces required to produce the movement [94, 95]. When two or more rigid segments are connected by joints, the whole system creates what is known as a kinematic chain [95]. There are two classes of kinematic chains: closed (Figure 2.4.7) and open (Figure 2.4.8). The difference between an open and a closed kinematic chain is that the rigid segments
Figure 2.4.7: An illustration of a closed kinematic chain connecting three rigid segments using three joints. The movements of the joint angles $\alpha$, $\beta$, and $\gamma$ are interconnected, and changes in one of the angles affects the others.

are joined in a closed circle with no freely-moving end, where an open kinematic chain contains a freely-moving end termed as the end effector.

In terms of human motion, the open and closed kinematic chain example is illustrated in Figure 2.4.9. An open kinematic chain example is the arm waving motion, where in Figure 2.4.9A the kinematic chain is created by the segment $x$-$y$-$z$, where the end effector that identifies it as an open kinematic chain is located at point $z$.

An example of a closed kinematic chain is shown in Figure 2.4.9B, where $a$-$b$-$c$-$d$-$e$-$f$ shows a closed kinematic chain formed by the legs and the ground by a squatting motion (i.e. the point $a$-$f$ where the foot are pressing on the ground created a closed kinematic chain). Similar to Figure 2.4.7, changes in one of the angles of a joint affects the angles of all the joints in the chain.

The two main area of Kinematics used in motion capture are:

- inverse Kinematics: Given a position of an end-effector (as illustrated in Figure 2.4.8), what are the angles ($\alpha$, $\beta$, and $\gamma$ in Figure 2.4.8) required for each segment. A short example on Inverse Kinematics on a two-dimensional plane is shown in Figure 2.4.10.

- forward Kinematics: Given the angles ($\alpha$, $\beta$, and $\gamma$ in Figure 2.4.8), what is the position of the end-effector relative to the origin. More details on Forward Kinematics is provided in Section 2.4.8.

A simplified version of Figure 2.4.8 is shown in Figure 2.4.10, where the segments are shown as $S_1$ and $S_2$ and an end-effector E located at a certain coordinates of X
Figure 2.4.8: An illustration of an open kinematic chain connecting three rigid segments and an end effector using three joints. The angles $\alpha$, $\beta$, and $\gamma$ are required to manipulate a robotic arm with known segment lengths so that the end of the arm (i.e. the end effector) is located at a desired coordinate with respect to the origin (i.e. the base of the arm). A change in one of the joint angles does not affect the others, unlike in a closed kinematic chain. [95].

and $Y$. The Inverse Kinematics process calculates the angles $\sigma$ and $\beta$ that is created by the two segments ($S_1$ and $S_2$) with the help of an imaginary line $L$ (created by drawing a straight line from the origin $O$ to the end-effector $E$). The line $L$ creates the angle $\theta$ which can help calculate $\sigma$ and $\beta$ using the law of cosines on the triangle $OJE$ in Figure 2.4.10:

\[
L = \sqrt{S_1^2 + S_2^2} \quad (2.4.1)
\]

\[
\theta = \text{atan}\left(\frac{Y}{X}\right) \quad (2.4.2)
\]

\[
cos(\alpha) = \frac{S_1^2 + L^2 - S_2^2}{2S_1L} \quad (2.4.3)
\]

\[
cos(\beta) = \frac{S_1^2 + S_2^2 - L^2}{2S_1S_2} \quad (2.4.4)
\]

\[
\sigma = \alpha + \theta \quad (2.4.5)
\]

where $\text{atan}$ is the inverse tangent function, and the angles $\alpha$ and $\beta$ can be calculated by using the inverse cosine function.
Figure 2.4.9: Human motion illustration of (A) open kinematic chain of arm waving, and (B) closed kinematic chain of a squat motion.

Figure 2.4.10: A simplified version of Figure 2.4.8 showing the Inverse Kinematics process on a 2D plane. The two segments are S1 and S2, with an end-effector E located at positions X and Y. The goal of Inverse Kinematics is to calculate the angles $\sigma$ and $\beta$. 
The segmented and hierarchical nature of the human body allow both kinematic techniques to be used to record human motion and reconstruct it using motion capture; after positional data of all the major joints in the body is captured (as in Figure 2.4.6) and the relationship between the joints are known (as in Figure 2.4.12), a model of the body can be constructed from captured positional data, where Inverse Kinematics is utilized to recover the angular values that are created between the joints, and Forward Kinematics is used to reconstruct the body from the angular values.

Why the need to express the human motion using angular values? The reason is to achieve flexibility in motion capture formats, and Kinematics describe the human body perfectly [95]. For example, to reach for a coffee mug, the brain calculates the required angles that needs to be created by the humerus and radius bones so that the hands arrive at the mug’s coordinate in space (given the knowledge of the brain owner’s length of the humerus and radius bones). Furthermore, absolute values of the joint positions effectively ties up a captured data to a specific scale and coordinate system, making the captured motion difficult to use in a system using a different scale. For example, an actor was captured by assuming that one unit in the coordinate system of the capturing system is one meter in length. If the captured motion was then used by an animator that creates his world by defining one unit in his coordinate system is one centimeter long (presumably to create a highly detailed world), then the actor effectively becomes a giant in the animator’s world. However, if the motion capture process relies on using the angular values that was created between bones, the animator can scale the captured model accordingly.

For the purpose of this Thesis, it is assumed that the motion capture data is already processed using Inverse Kinematics and the angular values between the bones are already known. The process of Inverse Kinematics is outside the scope of this Thesis. The process of Forward Kinematics will be described briefly in Section 2.4.8.

2.4.7 Motion Capture Representation of the Human Body

One of the more popular motion capture format is the Biovision Hierarchy (BVH) format, where the body is segmented into 38 bones as shown in Figure 2.4.11 which is the skeleton definition used in the Carnegie Mellon Motion Capture Database [96]. BVH is a popular format for motion capture due to the ease of which it can be read and reconstructed, since the information is stored in a plain text file. An example of the BVH format and its reconstruction process is detailed in Section 2.4.8.

The bones that are typically represented in motion capture are as follows:

- root: governs the movements of the body as a whole, usually located in the
Figure 2.4.11: Typical motion capture representation of the human body.
Figure 2.4.12: Typical motion capture skeleton hierarchy. The hierarchy shown in this Figure is a representation of the skeleton shown in Figure 2.4.11
• bones of the back: typically consists of the lower back, upper back, thorax, neck, and head.

• bones of the arms: consist of the humerus (upper arm), radius (forearm), and wrists. Some hierarchy also define the thumbs and the fingers.

• bones of the legs: consist of the femur (upper leg), tibia (lower leg), and foot. Some hierarchy also define the toes.

As shown in Figure 2.4.12, in general the body can be segmented into six major segments:

• the root,

• the left arm,

• the right arm,

• the left leg,

• the right leg, and

• the back.

where each segment consists of the individual bones that approximates the construction of the robotic arm shown in Figure 2.4.8, where the origin is the root for all segments, and the end effector is the extremities of the limb in question. For example, for the left leg in Figure 2.4.12, the origin is the root, and the end effector is the toes.

2.4.8 Motion Capture Reconstruction

From the stored angles in a motion capture file, the Forward Kinematics process described in Section 2.4.6 is used recalculate the original positions of the bones. This process is described in detail in this section.

How A Bone Is Described in Motion Capture

A body is described in motion capture using a series of local spaces connected together from end-to-end as shown in 2.4.13. The movements of the bone is described by rotation of the whole local space, and the physical length of the bone is described by using translation of a point from the origin of said local space. This concept is shown in two dimensions in Figure 2.4.13.
CHAPTER 2. MULTIMEDIA METADATA AND HUMAN MOTION

Figure 2.4.13: A simplified 2D illustration of global and local spaces that describes the movements of a bone. This example illustrates the movement of a bone described by the vector $\overrightarrow{LP}$ on a 2D plane, where $P$ is a point defined in the local space at coordinates $(X_{\text{local}}, Y_{\text{local}})$, $L_X$ and $L_Y$ describes the position of the local space in terms of the global space, and the local space is rotated (i.e. moved) $\theta$ degrees with respect to the global space. The goal of Forward Kinematics is to calculate the position of $P$ in terms of the global space coordinates.

In Figure 2.4.13, there are two spaces: A local space $L$ and a global space $G$, where $L$ exists within $G$. In the local space $L$, there exists two points: the origin of $L$ (shown as $L_x$), and an arbitrary point $P$. The line $\overrightarrow{LP}$ thus describes a bone, and the movements of the bone $\overrightarrow{LP}$ can be described by rotating the whole local space $L$.

This rotation of the whole local space $L$ is shown in Figure 2.4.13 as $\theta$. Since the location of the point $P$ remains constant in terms of the local space $L$, we need to calculate the position of point $P$ relative to the global space $G$ after $L$ is rotated $\theta$ degrees (as shown in Figure 2.4.13, where in Figure 2.4.13, the global position of point $P$ is shown as $X_{\text{global}}$ and $Y_{\text{global}}$).

The coordinates of $X_{\text{global}}$ and $Y_{\text{global}}$ can be calculated using a rotation matrix and taking into account the position of the local space inside the global space:
\[ P_{\text{global}} = R \cdot P_{\text{local}} + L_{\text{global}} \] (2.4.6)

where Equation 2.4.6 described the global coordinate of point \( P \) (\( P_{\text{global}} \)) that is rotated using the matrix \( R \) and translated using coordinate \( L_{\text{global}} \), and \( R \) is defined as:

\[
R = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}
\] (2.4.7)

For example, using the values shown in Figure 2.4.13, let \( X_{\text{local}} = 3 \), \( Y_{\text{local}} = 2 \), \( \theta = 30 \) degrees, and the local space \( L \) is located at coordinates \( L_{\text{global}} \) (which consists of \( L_x = 5 \) and \( L_y = 0 \)) in terms of the global space \( G \). Therefore, the position of point \( P \) in terms of global space \( G \) can be calculated as:

\[
P_{\text{global}} = \begin{bmatrix}
\cos(30) & -\sin(30) \\
\sin(30) & \cos(30)
\end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 0 \end{bmatrix}
\]

\[
P_{\text{global}} = \begin{bmatrix} 6.5981 \\ 3.2321 \end{bmatrix}
\]

Therefore, the point \( P_{\text{local}} = (3, 2) \), where the local space is rotated by \( \theta = 30 \) degrees and translated to \((5,0)\) with respect to the global space, can be described in global coordinates \( (P_{\text{global}}) \) at \((6.5981, 3.2321)\).

To simplify the calculation to be performed in Equation 2.4.6, the rotation and translation calculation shown in Equation 2.4.6 can be combined into a single step to create a transformation matrix which consists of both the rotation and translation of the local space. Let:

\[
L_{\text{global}} = \begin{bmatrix} L_X \\ L_Y \end{bmatrix}
\] (2.4.8)

\[
T = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & L_X \\
\sin(\theta) & \cos(\theta) & L_Y \\
0 & 0 & 1
\end{bmatrix}
\] (2.4.9)

where in Equation 2.4.8, \( L_X \) and \( L_Y \) denotes the position of the local space in terms of the global space. Equation 2.4.9 is a matrix that contains two distinct features: the upper left of the matrix is the rotation matrix from Equation 2.4.7, and the upper right is from Equation 2.4.8. The bottom row is part of a 3x3 identity matrix. The matrix \( T \) is therefore formed by:
Using the transformation matrix, Equation 2.4.6 can be simplified into:

\[
P_{\text{global}} = T \cdot P_{\text{local}}
\]  

(2.4.11)

since the rotation and translation step is now achieved using a single matrix \( T \).

Using the previous example of \( X_{\text{local}} = 3, Y_{\text{local}} = 2, \theta = 30, L_X = 5, \) and \( L_Y = 0, \)
the coordinates of \( X_{\text{local}} \) and \( Y_{\text{local}} \) would have to be converted into a 3x1 matrix to enable it to be multiplied with the 3x3 transformation matrix. This is achieved by putting the values of \( X_{\text{local}} \) and \( Y_{\text{local}} \) into a 3x1 identity matrix:

\[
P_{\text{local}} = \begin{bmatrix} X_{\text{local}} \\ Y_{\text{local}} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}
\]

and finally, the full \( P_{\text{global}} \) calculation becomes:

\[
P_{\text{global}} = \begin{bmatrix} \cos(30) & -\sin(30) & 5 \\ \sin(30) & \cos(30) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}
\]

By removing the identity portion of the result (i.e. the bottom row containing 1), the calculation gives an identical result to the previous example.

Since motion capture is inherently 3D (and the example above is only performing rotation and translation in 2D space), the calculations described above is performed three times (on the three separate planes of movement) for each bone in the motion data.

**Reconstructing The Movements of a Bone**

Throughout the rest of this Section, let \( \alpha \) denote the rotation in the \( x \) axis, \( \beta \) denote the rotation in the \( y \) axis, and \( \gamma \) denote the rotation in the \( z \) axis. In terms of motion capture, rotation in the \( x, y, \) and \( z \) axis represents movements in the Coronal, Transverse, and Sagittal planes, respectively.

From the known angles of a bone described in three planes of movements, a 3D
rotation matrix is constructed by taking the rotations in each of the plane separately, in the order prescribed in the motion capture data. Note that the order of rotation is not commutative, and to correctly reconstruct the motion, the order prescribed in the motion capture file must be strictly observed.

The 3D rotation matrix can be written as [95]:

$$ R = \begin{bmatrix} \cos X_x & \cos X_y & \cos X_z \\ \cos Y_x & \cos Y_y & \cos Y_z \\ \cos Z_x & \cos Z_y & \cos Z_z \end{bmatrix} \quad (2.4.12) $$

where the uppercase $X$, $Y$, and $Z$ denotes the axes of the global coordinate system, and the lowercase $x$, $y$, and $z$ denotes the axes of the local coordinate system.

Similar to the 2D case described in Section 2.4.8, a bone is described as a vector from the origin of its local space $L$ and a point $P$ where the vector $\overrightarrow{L_0P}$ describes the length of the bone. This location of the local space in terms of the global space in the 3D case is described using three variables of $L_X$, $L_Y$, and $L_Z$:

$$ L_{\text{global}} = \begin{bmatrix} L_X \\ L_Y \\ L_Z \end{bmatrix} \quad (2.4.13) $$

Since the calculation in Equation 2.4.6 would have to be performed for each bone and the motion capture skeleton is defined in a hierarchy of local spaces, the Equation 2.4.6 can be simplified using one 4x4 matrix (similar to the 2D scenario shown in Equation 2.4.9 and using a 3D expansion of the transformation matrix form from Equation 2.4.10) by concatenating the addition portion of $L_{\text{global}}$ (as described in [95]):

$$ T = \begin{bmatrix} \text{rotation} & \text{rotation} & \text{rotation} & \text{translation} \\ \text{rotation} & \text{rotation} & \text{rotation} & \text{translation} \\ \text{rotation} & \text{rotation} & \text{rotation} & \text{translation} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4.14) $$

where in Equation 2.4.14, $\text{rotation}$ denotes the 3x3 rotation matrix $R$ and $\text{translation}$ denotes the 3x1 matrix describing the coordinates of $L_{\text{global}}$. The resulting 4x4 matrix $T$ is known as a transformation matrix. Substituting Equations 2.4.12 and 2.4.13 into 2.4.14 results in:
Using this transformation matrix, the calculation for the position of $P$ becomes:

$$P_{\text{global}} = T \cdot P_{\text{local}}$$  \hspace{1cm} (2.4.16)

The global position of point $P$ can be calculated using Equation 2.4.6 albeit using a different rotation matrix $R$. In 3D space, the rotation matrix $R$ is defined as:

$$R = R_\alpha \cdot R_\beta \cdot R_\gamma$$  \hspace{1cm} (2.4.18)

where $R_\alpha$, $R_\beta$, and $R_\gamma$ are defined as:

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$  \hspace{1cm} (2.4.19)

$$R_\beta = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$  \hspace{1cm} (2.4.20)

$$R_\gamma = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (2.4.21)

Note that Equation 2.4.18 is only valid if and only if the order of rotation is $\alpha$, then $\beta$, then $\gamma$. For other order of rotation, Equation 2.4.18 would have to be rearranged to conform to the order prescribed. For example, if the order of rotation is defined as $\gamma - \beta - \alpha$, then Equation 2.4.18 would have to be rearranged into $R = R_\gamma \cdot R_\beta \cdot R_\alpha$.

Detailed information regarding how the angles are stored and reconstructed are presented in Appendix D for the interested readers.
2.4.9 The Human Body and Its Movement Descriptions: Summary

Different techniques in different area of studies have been representing the human body and its movements. The fields of anatomy, dance, and motion capture have their own separate representation of the human body, with different goals for each field: anatomy concentrates on divisions of the human body, dance notations concentrates on movements of the body, and motion capture concentrates on robust reconstruction of captured motion.

2.5 Conclusions

The explosion in multimedia content (evident by the traffic reported by video sharing sites such as Youtube [4], DailyMotion [12], and Vimeo [13]) made multimedia search and description a required technology for the future. MPEG recognized the need of a multimedia description scheme from as early as 2001 by standardizing the MPEG-7 Multimedia Description Schemes[16, 6], and similarly, Dublin Core standardized a simpler method to describe multimedia in the form of Dublin Core Metadata Initiative (DCMI) [7]. Multimedia description of content is desired, as evidenced by the popularity of an ad-hoc standard of ID3 [18] to describe MPEG Audio (MP3) [19] even without the backing of an international standard body.

Invariably, the multitude of multimedia content description standards also necessitates the creation of a flexible (and standardized) language to contain the description formats; the existence of which will enable a user to search and process multimedia that are described using different multimedia description standards. For example, if MPEG-7 and Dublin Core are described using the same language, a user can search multimedia described in both standards simultaneously. Extensible Markup Language (XML) [26] is currently the lingua franca of multimedia description formats; being human and machine readable [26], self-describing [41], transformable [29], and searchable [97, 98]. MPEG-7 is entirely based on XML, and Dublin Core is expressible in XML [53].

Although multimedia in the traditional sense (e.g. audio and video) are well represented in description area, in general there is a distinct lack of standard in describing human motion that is searchable (in the sense that MPEG-7 and Dublin Core are searchable). Ideally, a metadata of a human motion would be created either manually or automatically to describe the movements of the person that is structured and searchable, where the description is not too detailed as to include the idiosyncrasies of an individual’s movements but detailed enough to identify the moment of interest (e.g. when the right leg moves forward in a walking motion).
In other words, there is no computerized version of a movement notation such as Labanotation [14]. Variations of Labanotation expressed in XML existed in the form of LabanXML [86] and MovementXML [84]. However, searchability of motion is not the foremost goal of both works.

Therefore, this thesis attempts to answer the search and description of human motion by exploring: The communication aspect by examining and implementing the modern requirements of multimedia search (Chapter 3), the description aspect by the creation of a human motion description that describes human motion in a searchable manner (Chapter 4), and the extraction of the human motion description from 3D motion capture data along with an exploration in searching human motion (Chapter 5).
Chapter 3

A New Multimedia Query Format

Figure 3.0.1: The context of the Chapter in terms of the overall system: query communication (highlighted).

3.1 Introduction

With the amount of user-generated multimedia content uploaded into video sharing sites such as Youtube [1], Vimeo [2], Dailymotion [3], among many, the ability to be able to search those media by content is growing increasingly important. The latest statistics published by Youtube in 2012 [4] show that there are 72 hours of videos uploaded every minute, with 800 million viewers per month. The statistics also show that there are 3 hours of video uploaded every minute from mobile devices,
and 20% of the traffic to Youtube comes from mobile devices. The amount of video uploaded to such a popular video sharing site show that it is impossible to watch every single video uploaded to Youtube, therefore a search solution to reliably find a desired video would have to be developed. Video sharing sites typically employ a title-based and keyword-based (i.e. “tag”) descriptions of the uploaded video which were supplied by the user (i.e. uploader). However, these keywords are typically subjective and often ambiguous. For example, an uploader could tag a video as funny, while another person may not find it so.

MPEG recognized the need to describe multimedia by content, and in response developed the MPEG-7 Multimedia Description Scheme standard [6] to address this need. Similarly, Dublin Core Metadata Initiative also developed the Dublin Core standard [49] to describe multimedia by content, albeit not in such detail as the MPEG-7 standard (i.e. in Dublin Core, a media is described by its title, author, and other high-level informative items, while MPEG-7 allows low-level details such as color and spatial information to be described). However, there is currently a lack of a standard to describe human motion that is designed to be searchable. Survey conducted using 3,269,030 video uploaded to Youtube reveals that 22.9% of the video uploaded to Youtube were categorized as “music” (many of which potentially contain dance motions), and 9.5% of the video uploaded were categorized as “sport” (which also contain human motion) [8]. Being able to search the videos by motion would help refine the search results due to the amount of available videos in Youtube [4].

Searching human motion in multimedia is a problem that requires new solutions to be created (depicted in Figure 3.0.1), which includes: a new searchable description format to describe human motion; a new communication protocol to transmit those human motion descriptions that works with existing multimedia descriptions; and a method to extract the human motion descriptions.

If a user would like to search for “John Doe running while wearing a red shirt”, three things would have to be identified: John Doe, running, and a red shirt. The subject “John Doe” could be identified by a standard metadata such as MPEG-7 or Dublin Core (e.g. author is John Doe, subject is John Doe), and the red shirt can be identified using MPEG-7 color descriptors. However, the motion of running cannot currently be adequately described using existing multimedia metadata schemes. For example, how fast is John Doe running? Which leg does he use to start the running motion? etc. Currently, the associated detailed motion information would have to be described using a series of ambiguous keywords, and the number of possible keywords to describe a running motion can quickly escalate to an unmanageable size due to the different aspects of running that can be described (such as the speed, the starting leg, the number of steps, etc).
Exacerbating the problem is that there is no single querying method to unify all the popular multimedia description formats and a query communication format that handles all those description methods. This is reflected in the non-existent interoperability between popular media-sharing sites (such as Youtube [1], Vimeo [2], Dailymotion [3] etc). If a user would like to search for the “John Doe running while wearing a red shirt” video, the user would have to visit each site and execute the search using that site’s preferred description method. Search aggregation is one possible solution, however with no standard method of communication and result set presentation between client-to-server and server-to-server in multimedia query, search results from disparate sites would be difficult to unify.

The first step toward a solution to human motion multimedia search is therefore the creation of a new query format, which this chapter will propose. The goal of the new query format is twofold: to enable disparate multimedia description schemes in both large centralized multimedia database (e.g. Youtube) and personal multimedia database for private media sharing searchable using a single query. The second, equally important goal for the new query format is that it should also be able to contain a detailed description of human motion without resorting to ambiguous keyword-based descriptions.

For example, John Doe could have the option to upload to a video-sharing site his vacation video (which contains a video of him running while wearing a red shirt), or create his own database and share it personally while making it searchable to his select friends. In turn, John can also search Jane’s media database using the same query format. An illustration of such a scenario is shown in Figure 3.1.1.

In the area of multimedia search, MPEG recognized the lack of a standardized query format to perform a search on multimedia databases described using the MPEG-7 standard. To amend this shortcoming, MPEG (as the ISO/IEC SC29 WG11 working group) issued a call for proposal [99, 100] to create a new international standard to perform multimedia queries using MPEG-7. As one of the proposals submitted to MPEG during the MP7QF standardization process, the work presented in this chapter describes a proposed solution for a universal communication protocol between a client and a multimedia server, enabling the client to search for any multimedia content on any given server.

This chapter is organized as follows: Section 3.2 will provide the requirements of a general multimedia query format (where the requirements were defined by MPEG); Section 3.3 will detail the design of the proposed general multi purpose multimedia query format in the form of Multimedia Query Format (MQF) [101]; Section 3.5 will provide examples of the use of MQF; Section 3.6 will provide performance measurements of MQF performed during the MP7QF standardization process; Section 3.7 will explore the use of MQF in a stateful protocol environment; and finally Section
Figure 3.1.1: The general concept of a multi purpose multimedia query format, enabling a single query to be used to search large centralized multimedia databases and small personal multimedia databases.

3.9 will provide the conclusions of the chapter.

3.2 Requirements

The MPEG working group of ISO/IEC SC29 WG11 identified many capabilities that a modern multimedia query format is required to have to serve as an international standard. The result of the ISO/IEC SC29 WG11 working group was reflected in highly specific requirements in the MPEG-7 Query Format (MP7QF) call for proposal [99], which is organized within three aspects: the input and output query format requirements (described in Section 3.2.1), and query management requirements (described in Section 3.2.2).

To allow the goal of human motion centric multimedia search and to allow for multimedia queries using mobile devices, the requirements presented in this chapter defined one additional aspect: scalability requirements to allow MQF to include multimedia description schemes other than MPEG-7, and to allow MQF to work with a large centralized multimedia databases and small personal databases, with mobile devices with limited bandwidth acting as a client.
CHAPTER 3. A NEW MULTIMEDIA QUERY FORMAT

3.2.1 Query Format Requirements

This set of requirements [99] describes how the query format must be structured and describes what the input query format (the client’s query) and the output query format (the server’s reply) should be capable of:

1. **Include different methods of querying:** there are many methods beyond simple keyword search for multimedia, such as query-by-example, query-by-description, exact search by identifier etc. Query-by-example is a multimedia search concept that uses an example of a media to act as the query term in the search. For example, to search for a picture of a sunset, the user can use an example picture (see Section 2.3.2 in Chapter 2). The format would also need to search for unique identifiers such as International Standard Book Number (ISBN) for an exact match. It also needs to provide a method for traditional keyword-based search.

2. **Able to use any multimedia as a query term:** to facilitate query-by-example search method, the format needs to be able to provide a relatively simple method to allow a media to serve as the query term, either as an attachment in the query itself or a reference to an object available elsewhere.

3. **Able to mix any multimedia format in a single query:** other than being able to use a media as a query term, the format is also required to be able to include different multimedia formats to construct a query. For example, using a picture and a sound recording of a beach to return a video of a beach.

4. **Language and character set independent:** not all languages uses the English character set, therefore a universal multimedia query format would need to be able to use any language and character set.

5. **Able to construct complex query conditions using Boolean operators:** logical operators to describe the relationships between query terms is a basic required capability for a query format. For example, it should provide a method to allow the construction of a query such as “picture of a beach” and “sound of a wave” but not “picture of a car” to specify that the user would like a media of a beach without a car in it.

6. **Provide the ability to browse the contents of a multimedia database:** the format should allow a user to construct an empty query, e.g. to browse the contents of a multimedia database.

7. **Specifying the desired output media format/type of the query:** the format should, for example, allow the user to specify that the desired media...
is a video, a picture, a book, or any combination thereof. It should also allow
the user to directly specify the desired format, such as the format of a video
result (e.g. MP4, AVI, etc.).

8. **Specifying result limit and result paging**: the user should be able to
specify any paging requirements (e.g. 10 results per page) and result limiting
(e.g. return only the first 10 results). This is useful in a mobile environment,
where expensive bandwidth, screen size limitations, and memory limitations
are common.

9. **Structured result set**: the result set should be structured in a logical man-
ner. Descriptions of a result should be grouped accordingly. For example, all
descriptions of a returned video should be grouped together for ease of parsing.
This is to facilitate result set limiting and paging.

### 3.2.2 Query Management Requirements

This set of requirements [99] governs how query management tasks is to be handled
by the format:

1. **Exceptions specification**: there should be a standardized query error re-
porting specification that is clear and relatively easy to parse by a client with
limited processing capabilities.

2. **Querying server capabilities**: since the query format is a general format
that should work across different database implementations, users should be
able to discover what each server can provide. For example, one server with
sufficient processing power may be able to perform query-by-example, and
another server with limited processing power may only be able to provide
keyword matches.

3. **Relevance feedback**: for query-by-example, a user should be able to provide
additional information to a server whether parts of the returned results are
deemed relevant to the query or not. For example, a search of “beach with no
cars in it” might return instances of beaches with horses instead. The user
should be able to mark any or all instances of beaches with horses as irrelevant
and resubmit the query.

4. **Search within a previous result set**: related to the relevance feedback
requirement, a user should be able to specify that a search is to be done on
the previous result set returned by the server.
5. **Time limit for query responses**: time limit constraints for the query should be able to be specified by the user to limit waiting time in case the server is overloaded or the user queries many servers at once.

### 3.2.3 Additional Requirements

This set of requirements is an addition to the MPEG’s basic requirements previously described in Sections 3.2.1 and 3.2.2, and involves additional capabilities and use cases for a universal multimedia query format:

1. **Search aggregation**: the new format should enable aggregation of both the input query and the result set via a meta-search server, where the meta-search server could keep a central ranking system, grading multimedia database servers by comparing the database servers by the relevancy of results it returns. In a meta-search engine, there exists a ranking of the relevancy of a database’s response according to user supplied feedback. This relevancy ranking could be utilized for multiple databases hosting images, sound, video and textual data separately, grouped by media types. The database servers could then compete to provide the most relevant results.

2. **Personal multimedia databases**: on the other hand, the new format also has to be sufficiently flexible to allow it to be used on a personal multimedia database server which acts as both a client and a server. A user could have a special lightweight search engine which only indexes and stores image files, for example. A client utilizing the new format then could be used to search the personal database of images, or search any server on the Internet that is compliant using the same query.

3. **Stateful and stateless operation mode**: increasingly, mobile devices are being used as the primary method to access the Internet. Although current wireless technologies rival the speed of fixed-line connections such as Asymmetric Digital Subscriber Line (ADSL), by its nature wireless communication is inherently unstable, and therefore constant connection speed or availability of mobile devices cannot be assumed. A modern multimedia query format should allow for limited processing capabilities in both the client and the server (stateless operation, i.e. reducing the client and the server’s processing requirements), and a mobile user to communicate a query term-by-term for query refinement in an intermittent wireless connection (stateful operation, i.e. reducing bandwidth requirements).

4. **The use of multiple multimedia description schemes in a single query**: although the MPEG-7 standard encompasses many different aspects of a me-
dia (subjective, objective, low-level, high-level, etc.), it is not the only popular description standard. Dublin Core is another popular structured description standard, and user-supplied keyword-based descriptions are still widely used. Further, the query format should also be able to handle a new method to describe human motion (which is not standardized in both MPEG-7 and Dublin Core, and is described in detail in Chapter 4). Therefore, a query format that aims to unify multimedia search should also allow the use of other multimedia description standards and not place a limit on the scheme used to describe a media.

3.3 Design

To address the requirements and motivations described in Section 3.2, the Multimedia Query Format (MQF) is proposed. Key features of MQF are: the use of XML so as to render it human and machine-readable; the ability to handle complicated query conditions; the ability to work in a meta-search engine environment; the ability to work with any multimedia data; the use of both stateless and stateful mode of operation; and non-specified client and server technologies for future-proofing.

MQF was designed primarily to follow the Hypertext Transport Protocol (HTTP) [58] communication model, which is a widely used stateless communication protocol. Advantages of using a stateless approach for search include relatively simple client and server implementation compared to a stateful protocol, since each query is considered a standalone query unconnected to other subsequent queries from the same client. A stateless approach therefore allows the server to free up resources as soon as a reply is sent to the client.

3.3.1 XML-based Query Format

XML is a standard data interchange method used in many standards such as MPEG-7, MPEG-21, SOAP, WSDL, Dublin Core, etc. Using XML also allows MQF to leverage existing XML processing tools available in many programming languages.

The use of the XML standard also allows MQF to be easily expandable to use any international standards for multimedia description that utilize XML technology, such as the aforementioned MPEG-7, MPEG-21 and the Dublin Core metadata initiative. Furthermore, any standard defined in MPEG-7/21 (such as user preferences) can be communicated to the server by using MQF as an envelope.

For presentation of the result set from a query, a client or a server can transform the resulting XML formatted reply by using XSLT, hence minimizing the effort required for presentation. The use of XML technology directly provides MQF with
language and character set independency, since XML was designed to be a general format for data interchange.

3.3.2 The use of Reverse Polish Notation (RPN) for Query Logic

Logical operators to describe the relationships between query terms is a required capability for a query format, (which was reflected in MPEG’s requirements described in Section 3.2.1) where the query terms are connected using Boolean logic to describe that the client wants the result set to be based on the query terms. For example, to search for a picture of a beach and the sun, the query terms can be written as “beach AND sun”. Conversely, if the client wanted to search for a picture of a sun without a beach, the query terms can be written as “sun AND NOT beach”.

Although standard mathematical notation can be used to encode complex query logic such as “(A OR B) AND C”, the use of parenthesis to determine the order of operation creates more complication for logic processing. RPN allows an equation (and thus query logic) to be communicated without the use of parenthesis, which simplifies communication since RPN is processed in a stack-based manner. As precedence in query format implementations, RPN is widely used in the Z39.50 standard [102] which was originally designed for library search.

By substituting numerical operands with any multimedia content as in MQF, any server is then able to perform very complex operations based on multimedia data unambiguously and without the need to signal which operand takes precedence over which. The server is able to parse the query input in sequence, hence minimizing the memory requirements while at the same time ensuring the accuracy of the output.

3.3.3 Query Abstraction Levels to Minimize Ambiguity in Query Terms

To facilitate unambiguous interpretation of a query, the concept of query abstraction levels is introduced here. There are four levels of query abstraction:

1. **Exact level** (“**exact**”): looking for an exact match; could be based on the descriptors provided. E.g. MPEG-7 descriptors, Dublin Core metadata.

2. **Query by example level** (“**example**”): looking for a similar match based on a given example.

3. **Semantic description level** (“**semantic**”): looking for a match by describing the semantics of the media. An example of a semantic description is Concept Languages [103] and Labanotation [78] for human motion.
4. **Natural language level** ("freetext"): looking for a match by vague, natural language description of an event or a multimedia item.

From the “exact” level to the “freetext” level, the query becomes more difficult to match exactly and requires more computations to process. For example, searching for the keyword of “John” involves searching the database for media described using the word “John” without regard to the context to the keyword (i.e. exact match level). On the other hand, searching using a natural language sentence such as “What did John do last summer” involves determining the context of the word “John” in the sentence (i.e. natural language level), hence requiring more resources to process.

The query abstraction concept is required to instruct the server how to interpret a certain query term and what kind of match is desired by the client. The abstraction level information is a required attribute for any query term to avoid interpretation confusion by the server.

### 3.3.4 Design: Conclusions

The combination of Query Abstraction Level and the use of RPN notation provides the ability of MQF to be as free form as possible, where a query can consist of multiple media formats, in different levels of abstraction and combined with a complicated query condition. The use of XML allows the multitude of currently available XML processing tools to be used to process MQF, and also allows little to no modification of MQF to incorporate different multimedia description formats that are either based on XML or can be represented in XML (e.g. MPEG-7, Dublin Core) to be used as MQF query terms.

Note that MQF is a format specification and does not specify how a server should process a query except that the default response to the client should be in XML format. MQF is concerned only with how a client should formulate a query, resulting in flexibility on the server side. A server, for example, can choose not to process a particular query abstraction level due to processing power or other limitations. This flexibility allows advancement in technology to be implemented immediately on the server side without any impact on the communication protocol between the client and the server.

### 3.4 MQF Schema Descriptions and Example Instantiations

The Multimedia Query Format (MQF) XML Schema is an implementation of the design outlined in Section 3.3 and follows the requirements outlined in Section 3.2.
The MQF implementation consists of four major parts:

1. The main `<mqf>` element which encapsulates all other MQF elements.

2. The `<query>` element which contains query terms encapsulated in `<item>` elements, and `<operator>` elements which specifies the logical operation to be performed on the query terms (i.e. `<item>` elements).

3. The `<replies>` element which encapsulates the server replies. Each reply item is encapsulated inside a `<reply>` element.

4. Query management elements, which consist of elements for service discovery and error reporting.

In MQF, each query is by default a stateless query, i.e. each query stands alone with no connection between former or subsequent queries even from the same client. This design decision is made to emulate the workings of HTTP and to minimize server hardware/performance requirements. A stateful approach to queries using MQF, however, is an additional requirement for using MQF in a mobile environment (described in Section 3.2.3). Stateful operation of MQF is further discussed in Section 3.7.

### 3.4.1 The root `<mqf>` element

The main `<mqf>` root element encapsulates all other MQF elements, and only one `<mqf>` element can be present in a single query. The root `<mqf>` element is illustrated in Figure 3.4.1 where it specifies two attributes:

- “version” that specifies the MQF version (in this Thesis, the version attribute is always set to “1.0”).
- “xml:lang” that specifies the natural language the query is using as defined by RFC 1766 [104]. For example, `xml:lang="en"` for English, `xml:lang="fr"` for French.

The root `<mqf>` element contains children nodes of:

- the `<queryId>` element that is a unique identification string of a query. There is no restriction on what string can be used for the query ID. For example: `<queryId>Query1</queryId>`

- the `<from>` element that specifies the URI of the client. For example:

  ```
  <from>localhost</from>
  ```

  which signifies that the query originated from the local machine and directed toward a server located in the same machine, or
In due course, we should install the relevant ISO 2- and 3-letter codes as the enumerated possible values...

Figure 3.4.1: The root <mqf> element illustration.
Figure 3.4.2: Example MQF instantiation of the root `<mqf>` element that contains the elements `<queryId>` and `<from>`, and attributes of “version” and “xml:lang”. Note that this instantiation is not yet well-formed since the required sub-elements (one of `<query>`, `<replies>`, `<exceptions>`, or `<services>`) are not present.

- `<from>192.168.1.1</from>` which signifies that the query originated from a remote machine.

For more information on URI, see Chapter 2 Figure 2.2.10

- one of the main query operation elements of MQF, which consist of:
  - `<query>` if a query is to be performed,
  - `<replies>` if a query reply,
  - `<exceptions>` if an error occurred, and
  - `<services>` for a service discovery request reply from the server.

Note that the four elements are mutually exclusive. For example, in a single MQF document, there cannot be both `<query>` and `<reply>` elements occurring.

An example instantiation of the root `<mqf>` element is depicted in Figure 3.4.2, where in Figure 3.4.2, the `<queryId>`, `<from>`, and both attributes (version and xml:lang) are present. Note that the example in Figure 3.4.2 is not yet a well-formed XML since the required `<query>`, `<replies>`, `<exceptions>`, or `<services>` is not present.

### 3.4.2 The `<query>` and `<item>` elements with RPN format

The existence of a `<query>` element signifies that the MQF is in a “query mode”, that is, the contents of the MQF document is to be treated as an input query. On each MQF document, only one `<query>` element is allowed to be present. The `<item>`
The <query> element

The illustration of the <query> element is shown in Figure 3.4.4, where it contains reply format specification, query terms (“items”), and operators.

The reply format specifications consists of:

- <replyType> that specifies the desired type of media using the standard Multi-purpose Internet Mail Extensions (MIME) specification, which is described in detail in RFC 2046 [105]. It is possible to specify more than one <replyType> element if the client desired more types of media to be searched. For example:

  <replyType>image/jpeg</replyType>

to specify that the client desired images encoded in jpeg,

  <replyType>video/mp4</replyType>

for video media encoded in MPEG-4 format.

  <replyType>audio/**</replyType>

specifies that the user would like to search for any type of audio.
Figure 3.4.4: The <query> element illustration.
• `<replyAs>` specifies the XPath [98] of the matching media’s description to be returned (if the description format is known beforehand by the client). The `<replyAs>` element provides a shortcut for the client to return exactly the description required without having the server to send the whole description to the client in an effort to save transmission bandwidth, but this feature requires the database to be described in a certain description format, such as MPEG-7. For example:

```xml
<replyAs>//mpeg7:CaptionLanguage[text()="en-US"]</replyAs>
```

leveraged MPEG-7’s capability to describe the media’s closed-caption language. This example specifies that the user would like the server to return medias and their closed-caption language should be in US English.

• `<startIndex>` and `<numOfResult>` specifies the required result set paging information to be transmitted. For example:

```xml
<startIndex>1</startIndex>
<numOfResult>10</numOfResult>
```

specifies that the client would like the result starting from result index of 1 with 10 results (results number 1 to 10).

```xml
<startIndex>11</startIndex>
<numOfResult>10</numOfResult>
```

fetches the second page of the result set (results number 11 to 20).

• `<sortBy>` and `<groupBy>` specifies the sorting and grouping (similar to sorting and grouping in SQL) of the result set. For example:

```xml
<sortBy>//dc:title</sortBy>
```

specifies that the result set should be sorted according to the title, alphabetically.

```xml
<groupBy>//dc:creator</groupBy>
```

specifies that the result set should be grouped according to the author of the media.

• `<timeOut>` specifies the time limit of the query in seconds. For example:

```xml
<timeOut>60</timeOut>
```

specifies that the query should be attempted for 60 seconds. Otherwise, the client (and the server) would consider the query to be canceled.
The `<item>` element

The `<item>` element encapsulates each query term. The illustration of the `<item>` element is shown in Figure 3.4.5. An `<item>` element can contain any description fragment in the MPEG-7, Dublin Core, or Human Motion Markup Language (HMML, see Chapter 4) format.

Additionally, the `<item>` element is required to specify the “queryLevel” attribute (as per the requirement in Section 3.3.3). Possible values are:

- **exact**: signifies that the item is representing an exact description. For example:

  ```xml
  <item queryLevel="exact">
    <dc:creator>Neal Stephenson</dc:creator>
  </item>
  ```

  means that the user would like to search for a media authored by Neal Stephenson.

- **example**: signifies that the item is an example. The item involved could be an attached binary, and can also signifies that the query term is not supposed to be matched literally, but the user would like to search for a similar result. For example:

  ```xml
  <item queryLevel="example">
    <mpeg7:MediaUri>http://server/image.jpg</mpeg7:MediaUri>
  </item>
  ```

  signifies that the linked image is an example for the desired match.
Figure 3.4.6: The <replies> element illustration. The element encapsulates all individual match items in the result set, each in a separate <reply> element. A result set can also contain a general description of the set in a textual format, which is encapsulated in a <text> element.

```xml
<item queryLevel="example">
  <dc:title>World Cup</dc:title>
</item>
```

signifies that the title “World Cup” is an example, and the returned media may not strictly be associated with the World Cup, but soccer in general.

- **semantic**: signifies that the item is described using a semantic description language such as Human Motion Markup Language (HMML), which will be presented in Chapter 4.
- **freetext**: signifies that the item is a natural language description of a media. For example:

  ```xml
  <item queryLevel="freetext">
    penalty kick scoring a goal
  </item>
  ```

signifies that the user would like to search for a media containing a successful penalty kick.

### 3.4.3 The <replies> and <reply> elements

The <replies> element signifies that the MQF document contains a result set that was constructed by the server in reply to a client’s query. The illustration of the content of the <replies> element is shown in Figure 3.4.6. Each result item that is returned by the server is encapsulated inside a <reply> element, the illustration of which is shown in Figure 3.4.7.

The structure of the <reply> element (in Figure 3.4.7) is similar to the <item> element in query mode shown in Figure 3.4.5, with some differences:

- the matching item is encapsulated in the <item> element, which contains reference to the matching item. The reference to the media could be represented
in two forms: as a link (using MPEG-7’s MediaUri element); or as a binary attachment in Base64 encoding (using MPEG-7’s InlineMedia element).

- the <description> element which describes the item in MPEG-7, Dublin Core, or Human Motion Markup Language. The <description> element also contains a <thumbnail> element which contains a link to a thumbnail of the matching item. The definition of the <description> element is shown in Figure 3.4.8. Additional multimedia description schemes can be included by adding the relevant namespace in the MQF XML schema.

- the <index> element which specifies the result item’s index number in the result set.

### 3.4.4 Service discovery element

Since MQF was designed as a communication method between disparate multimedia database servers, potentially different capabilities among those databases is to be expected and cannot be assumed. For example, one multimedia databases might exist on a very powerful hardware that can perform fast query-by-example using natural language processing. However, a personal multimedia server contained within
In due course, we should install the relevant ISO 2- and 3-letter codes as the enumerated possible values.

In order to allow MQF to take into account different server capabilities, a mechanism for discovering a server’s capabilities is built into MQF using the `<services>` element. The definition of the `<services>` element is shown in Figure 3.4.9, and a client discovers a server’s capabilities by transmitting an empty `<services>` element.

The server reply of a service discovery request is encapsulated in a `<services>` element, which contains:

- **queryLevel**: returns the server’s supported query level. For example:

  ```xml
  <queryLevel>exact</queryLevel>
  <queryLevel>example</queryLevel>
  ```

  shows that the server is able to process query levels of exact and example, and incapable of processing semantic and freetext levels.

- **version**: returns the server’s supported MQF version.

- **replyType**: returns the server’s media types in MIME format. For example:

  ```xml
  <replyType>image/jpeg</replyType>
  <replyType>video/mp4</replyType>
  ```

  shows that the server contains two types of media: images in JPEG format and videos in MPEG-4 format.
• descriptionNamespace: returns the XML description namespace (in URI format) of the database content. For example:

    <descriptionNamespace>
    http://purl.org/dc/elements/1.1/
    </descriptionNamespace>

shows that the server’s media description scheme is using Dublin Core, and

    <descriptionNamespace>
    urn:mpeg:mpeg7:schema:2001
    </descriptionNamespace>

shows that the server is using MPEG-7 as a media description scheme. Note that it is possible for a server to describe their contents using both description formats.

• operator: returns the operators supported by the server. For example:

    <operator>AND</operator>
    <operator>OR</operator>
    <operator>NOT</operator>

shows that the server can apply AND, OR, and NOT logical operators to the query terms. Since the operators are not defined in the MQF schema, this step is important to know what operators are supported by a server (which could be different from server to server) so that the client can construct a query using operators that that particular server understands.

If a server describes a multimedia item using two different standards (e.g. using both Dublin Core and MPEG-7 to describe the same item), a mapping between equivalent information such as creator information (<dc:Creator> in Dublin Core and <mpeg7:Creator> in MPEG-7) would have to be created in the server. The actual mapping of equivalent information between metadata standards is not within the scope of this thesis.

3.4.5 Exceptions

In the occurrence of an error either from the client side (i.e. non well-formed query, requesting invalid media type that is not available on the server), or the server side (i.e. internal server error), MQF returns HTTP-compatible error messages (listed and maintained by the Internet Assigned Numbers Authority / IANA, available in [106]). Example MQF instantiation of an exception is shown in Figure 3.4.11.
Figure 3.4.10: The <exception> element definition which contains the exception code and textual description of the error. The <code> element contains an HTTP-compatible error codes along with a textual description of the error inside the <description> element.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<mqf xmlns="urn:uow:mqf:schema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:uow:mqf:schema file:mqf.xsd">
  <queryId>1</queryId>
  <from>server</from>

  <exceptions>
    <exception>
      <code>503</code>
      <description>Service Unavailable</description>
    </exception>
  </exceptions>
</mqf>
```

Figure 3.4.11: An example instantiation of a server exception using HTTP code 503 (service unavailable). This error could signify that the server is down for maintenance.
3.5 MQF Example Workflow

Figure 3.5.1 shows a typical operation using MQF, where a query is started by performing a service discovery request to a server, and sending the query. The server then responds with a list of media from its database that matches the query, or an exception if an error has occurred.

For example, a user would like to search for “a video of John Doe running and jumping into a lake”. The progression of the query using MQF is:

1. service discovery (client to server)

   Before any query can take place, the client must discover what the server is capable of. The client initiates connection using an empty `<services>` element (Figure 3.5.2).

2. service discovery reply (server to client)

   The server replied with available query levels, available media types, and allowed logical operators (Figure 3.5.3). The service discovery reply shows that
the server is capable of processing exact queries and query-by-example, but
no capabilities for semantic queries nor freetext queries. The server contains
three types of media: images, audio, and video with descriptions in Dublin
Core format. It also able to apply logical operators of AND, OR, and NOT to
incoming query terms.

3. query (client to server)

Using RPN notation, the client performs a query of “search for video where
the subject is John Doe running and jumping into a lake” using Dublin Core
descriptions, since the server indicates that its contents are described using
Dublin Core in the previous step (Figure 3.5.4).

4. replies (server to client)

The server replies (in Figure 3.5.5) that in its database, there are two items
that match the client’s query. The first item is a video by John Doe about him
running and jumping into a lake, and another video was recorded by Jane Doe
(recorded from a different point of view). The reply items contains the URI
of the video on the server (http://server/john.mp4 for the first match, and
http://server/jane.mp4 for the second match). Both items are transmitted
along with the relevant descriptions in Dublin Core format.

5. exceptions (server to client)

If the client’s query is not well-formed or damaged, or if the server is down
for maintenance, exceptions can be generated and transmitted to the client
using MQF. In the example shown in Figure 3.5.6, the server replies with an
HTTP-compatible error code of 400 (bad request) that informed the client
that the query was not well-formed. Possible causes are corruptions during
transmission, or the MQF query was incomplete.
<?xml version="1.0" encoding="UTF-8"?>
<mqf xmlns="urn:uow:mqf:schema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:uow:mqf:schema file:mqf.xsd">
  <queryId>1</queryId>
  <from>server</from>

  <services>
    <version>1.0</version>
    <queryLevel>exact</queryLevel>
    <queryLevel>example</queryLevel>
    <replyType>image/jpeg</replyType>
    <replyType>audio/mp3</replyType>
    <replyType>video/mp4</replyType>
    <descriptionNamespace>
      http://purl.org/dc/elements/1.1/
    </descriptionNamespace>
    <operator>AND</operator>
    <operator>OR</operator>
    <operator>NOT</operator>
  </services>
</mqf>

Figure 3.5.3: The server’s service discovery reply showing the query level capabilities of the server, media types available in the database, and allowed logical operators. This example shows that the server is capable of performing exact match and query-by-example (but not semantic nor freetext queries), contains images (in JPEG format), audio (in MP3 format), and video (in MP4 format), and supports AND, OR, and NOT operations. The presence of Dublin Core description namespace (“http://purl.org/dc/elements/1.1/”) signifies that the server’s contents are described using Dublin Core.
<?xml version="1.0" encoding="UTF-8"?>
<mqf xmlns="urn:uow:mqf:schema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:dc="http://purl.org/dc/elements/1.1/">
    <queryId>1</queryId>
    <from>client</from>
    <query>
        <!-- Requested media is MP4 video -->
        <replyType>video/mp4</replyType>
        <!-- The subject of the video is John Doe -->
        <item queryLevel="exact">
            <dc:subject>John Doe</dc:subject>
        </item>
        <!-- ...running... -->
        <item queryLevel="exact">
            <dc:description xml:lang="en">run</dc:description>
        </item>
        <operator>AND</operator>
        <!-- ...and jumping... -->
        <item queryLevel="exact">
            <dc:description xml:lang="en">jump</dc:description>
        </item>
        <operator>AND</operator>
        <!-- ...into a lake. -->
        <item queryLevel="exact">
            <dc:description xml:lang="en">lake</dc:description>
        </item>
        <operator>AND</operator>
    </query>
</mqf>

Figure 3.5.4: The client performs a query of “search for a video where John Doe is running and jumping into a lake” using Dublin Core descriptions as part of the query terms.
Figure 3.5.5: The server replies with two relevant matches in its database.
Figure 3.5.6: If the client’s request is not well-formed XML or unrecognizable by the server, the server will reply with HTTP-compatible status code. In this example, the status code returned is 400 (Bad request) due to the error in the client’s formulation of the query.

### 3.6 Testing and Measurements

This section discusses the testing and measurements performed by MQF during the MPEG-7 Query Format standardization process to verify the conformance of MQF to the requirements put forth by MPEG.

#### 3.6.1 MPEG Call for Proposal Conformance Testing

As part of the standardization process in MPEG, proposals for the MP7QF standard are tested using a set of use cases [107, 108, 109, 110]. The use cases are agreed upon by the standardization participants which include simple queries, complex queries, free text queries, client request for specific output formats, and server replies (which include exceptions and error messages) to confirm the conformance of the proposal to the requirements put forth by MPEG. The test set are:

1. 18 basic queries using a single query item.
   
   Example: “Return images like the attached one. The image is located in the user’s hard drive and not located in any server.”

2. 24 complex queries using many items and operators.
   
   Example: “Return images of either a mountain or a beach with a car or a motorcycle in the image but not both.”

3. 15 free text queries.
Example: “A query for images of the winners of the last ten Australian Open tennis tournament.”

4. 25 queries involving the client requesting a specific output format.

Example: “Return all images with the creator information, creation date, image format and a thumbnail of the image.”

5. 14 server replies.

Example: A server reply to any of the above queries.

For each of the queries, the following measurements were taken: size of the XML file in Bytes; number of elements and attributes of each XML; memory used for parsing the XML using DOM in Kilobytes (1024 bytes); and speed of parsing in milliseconds.

The test for MQF was performed using a Pentium Core 2 Duo 2.13 GHz machine running Windows XP SP2 with 2 GB of RAM. The parser is written using Python 2.5. The results of the measurements are presented in Table 3.6.1.

3.6.2 Results

Table 3.6.1 details that the average parsing performance of MQF in terms of time and memory used by the XML parser on average is stable (i.e. complex queries do not significantly increase resource usage compared to simple queries), even though the actual XML instantiation itself contains significant differences in terms of the number of elements and the file size.

This resource usage stability is an advantage of MQF, as the parsing requirements for RPN queries (which is specific to MQF) only mandates a server processing the query to process it in-order. The use of RPN combined with the relatively stable memory and time parsing requirements of MQF results in a stable query format overall in terms of processing requirements.

3.7 Query Streaming

To perform multimedia queries using MQF in a stateful manner, MQF includes a “Query Streaming” (i.e. stateful) mode. Similarly, HTTP allows a stateful information interchange by using “cookies”, which is standardized in RFC 6265 – HTTP State Management Mechanism [111]. Although there are advantages in using a stateless design (as outlined in Section 3.3), stateless method of communication allows a series of queries to be communicated relatively efficiently compared to a stateless approach (since the client and server “remembers” all preceding communications).
In contrast to a stateless approach, a stateful approach would require the server to keep the state of a query in memory until either the client disconnects or declared that the query is finished, preventing the server to free any resources in the meantime. However, there are advantages to the stateful approach, one of which is the server “knowing” the context of subsequent queries, which is a more natural method of communication between people. For example, it is arguably more natural to have this conversation:

Q: *Is John having a holiday?*
A: *Yes he is.*
Q: *With Jane?*
A: *Yes.*
Q: *On a beach?*
A: *Yes.*

Rather than a more “stateless” conversation such as:

Q: *Is John having a holiday?*
A: *Yes he is.*
Q: *Is John having a holiday with Jane?*
A: *Yes.*
Q: *Is John having a holiday with Jane on a beach?*
A: *Yes.*

To allow MQF to perform a more conversational, stateful search, the concept of “Query Streaming” is introduced in this Section.
CHAPTER 3. A NEW MULTIMEDIA QUERY FORMAT

3.7.1 Introduction to MQF Query Streaming

Let us consider a query-by-example scenario where the user is searching for “a picture of a beach containing a sunset”:

1. the user sends the initial query using keywords: “beach” and “sunset”.

2. the server returns all pictures identified by the two keywords.

3. the user identifies a number of pictures that he/she does not like and resends the refined query to the server.

4. the server filters out the result set based on the user’s feedback and resends the trimmed result set to the client.

5. the process is repeated until the user finds the desired picture.

An illustration of this process is depicted in Figure 3.7.1. This multi-step process allows the client to perform a logically complex query using a relatively simple process. During the feedback phase of the query (steps 3-4), the server could determine the characteristics of the images that the user removes using any method, examples of which could include identifying other keywords associated with the removed picture, MPEG-7 color histogram description of the removed pictures, etc.

On a mobile device with unpredictable wireless connection, it is not feasible to perform this type of query using a stateless system such as HTTP, e.g., the
client would have to send and re-send every refinement as a standalone query. This increases the mobile device’s processing power usage during a query, and puts the burden of identifying the characteristics of the pictures that the user removes into the mobile device instead of the more powerful servers.

As a solution, stateful MQF communicates the query piece-by-piece using Fragment Request Unit (FRU) and Fragment Update Unit (FUU), both are part of MPEG standards. The server could keep a continually-updated MQF document, with the client’s refinements sent to the server not as a new MQF document, but as an XML fragment that would update the server’s view of the client’s MQF query.

This method of communication using XML fragments is used in the Extensible Messaging and Presence Protocol (XMPP) [38], where two communicating parties keep separate XML documents that is continuously being synchronized using XML fragments. MQF with FRU/FUU emulates this concept in a multimedia search environment.

3.7.2 Using Fragment Request Unit (FRU) and Fragment Update Unit (FUU) to operate Query Streaming

Using both FRU and FUU (described in Chapter 2), it is possible to transmit fragments of an XML document. Another capability of FRU and FUU is to synchronize two documents in a piece-by-piece basis. By exploiting this capability of FRU and FUU, it is hence possible to synchronize the contents of two MQF documents, and thus achieve a stateful MQF query. The workflow of MQF Query Streaming using FRU and FUU is therefore as follows:

- all the query and replies from both the server and the client are performed using MQF. The client and the server each has a copy of the client’s MQF query, with the server’s copy initially blank.
- from the initial query, the server constructs a result set which is transmitted to the client.
- additional search terms are sent to the server using FUU to update the server’s copy of the query.
- the server transmits the updated result set to the client. Note the result set does not necessarily contains only additions, since it is possible that the additional query terms from the client results in the removal of some items from the result set.

The workflow of MQF Query Streaming described above is illustrated in Figure 3.7.2.
Figure 3.7.2: Workflow of stateful MQF multimedia query using Query Streaming. Both the client and the server possess a copy of the client’s query and the result set. Both documents are synchronized using FRU and FUU.
CHAPTER 3. A NEW MULTIMEDIA QUERY FORMAT

The primary advantage of this method is the elimination of the need for the user to examine each result and indirectly eliminating vague query keyword by guiding the user through the description of the result set. This scenario is highly desirable to enable multimedia query in a limited bandwidth and/or unreliable network environment where the bandwidth cost is prohibitive or the data transmission rate is highly variable.

Additional reduction in bandwidth usage required in transmitting MQF queries could be achieved by using BiM (in Chapter 2), which is a technique to represent a textual XML document using a binary representation. In contrast to compression algorithms such as Lempel-Ziv [44], BiM does not compress the XML document which would require decompression back to textual representation before any XML processing took place. Instead, BiM leverages the information that is present in the XML schema that describes the structure of the document and replaces textual representation of nodes and data using a shorter binary representation while retaining the structure of the document in a shorter binary form. By using BiM in conjunction with the MQF XML schema, significant reduction in the XML file size can be achieved, which will be further analyzed in Section 3.7.3.

3.7.3 MQF Query Streaming Scenarios

In the scenarios presented in this Section, two sets of results are presented: using plain text-based XML format and a binary-compressed BiM (Binary MPEG format for XML) which is part of the MPEG-B standard ISO/IEC 23001-1 [43].

Scenario 1: Image query returning thumbnails

As an illustration, consider a scenario where the user wants to search for a picture of a mountain, with some vague idea of what kind of mountain he/she wants to see. Note that this example was actually performed using a popular search engine, where the first page of results is taken as the data for this example.

1. The initial search term “mountain” is sent to the server to initiate the query.

2. The server queries the database for any image with the keyword “mountain” which results in 20 items, and sends the associated keyword metadata contained in the result set to the client, for example: Mountain range, desert, snow, lake, river, animal, movie and news.

3. On the keyword list, the user then discovers that a mountain covered in snow is the desired result, which in this example results in only three out of twenty results in the result set.
Search for a picture of a mountain

20 matching results found on server

What are other descriptions of those pictures

Mountain ranges, snow, desert, etc.

Mountain with snow

Three matches found

Initial query

Server reply

FRU request

FUU reply

Refine query

Server reply

Result set

Updated result set

Figure 3.7.3: Image query scenario using Query Streaming. The client requested pictures of a mountain (1), where there are 20 matching images with the keyword “mountain” in the server (2). The client instructs the server to provide additional keywords associated with those pictures (3) and the server sends them to the client (4). The client adds an additional term of “snow” to the original query (5), and the server replies that there are only three pictures that uses both “mountain” and “snow” keywords (6).
4. Step 2 is repeated as many times as necessary to further refine the query.

Figure 3.7.3 shows the flow of communication between the client and the server in this scenario.

The total data transmitted to the client in this scenario is shown in Tables 3.7.1 to 3.7.4. Tables 3.7.1 and 3.7.2 is for the scenario where no Query Streaming is performed and all thumbnail results are returned to the client from the server. The total size of the reply is the sum of the thumbnail data files (as a binary attachment) and the additional MQF XML file.

<table>
<thead>
<tr>
<th>XML file (without Query Streaming)</th>
<th>Original size (bytes)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total size</td>
<td>Binary attachment size (text format)</td>
</tr>
<tr>
<td>Server reply: 20 embedded thumbnails</td>
<td>114,743</td>
<td>97,979</td>
</tr>
</tbody>
</table>

Table 3.7.1: Image query scenario without Query Streaming, uncompressed.

<table>
<thead>
<tr>
<th>XML file (without Query Streaming)</th>
<th>Compressed size (bytes)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total size</td>
<td>Binary attachment size (binary format)</td>
</tr>
<tr>
<td>Server reply: 20 embedded thumbnails</td>
<td>74,376</td>
<td>73,484</td>
</tr>
</tbody>
</table>

Table 3.7.2: Image query scenario without Query Streaming, BiM compressed.

Using BiM, the total MQF document size to 5% of its original size (without Query Streaming) and 10% of its original size (total transmitted size with Query Streaming). Compared to the scenario where Query Streaming and BiM compression is not performed, results of Table 3.7.3 show the total files size of the server replies is reduced to approximately 10% when compared to the results of Table 3.7.1 without BiM compression.

The results in Tables 3.7.1 to 3.7.4 show that while using Query Streaming can significantly reduce transmission bandwidth, individual application would need to implement a proper strategy if bandwidth reduction is the primary aim for that particular application, as Query Streaming involves multiple communications between client and server (instead of a single query and a single reply without Query Streaming) which could potentially increase the total amount of data to be transmitted if
CHAPTER 3. A NEW MULTIMEDIA QUERY FORMAT

many query refinements are made. The significant reduction in total transmitted bytes from Table 3.7.2 (compressed, without Query Streaming) to Table 3.7.4 (compressed, with Query Streaming) stems from the lack of need to transmit all 20 thumbnails and utilizes Query Streaming to determine that only 3 thumbnails needs to be sent to the client. The results show that with or without Query Streaming, using BiM compression on MQF reduced its size significantly.

Scenario 2: Audio Query

In this scenario, the user would like to search for a song using query-by-example. The actual steps of the query are shown in Figure 3.7.4:

1. the user sends a sample of a song to the server.

2. the server replies that it has 100 possible matches for the song since the ex-
Figure 3.7.4: Audio query scenario using Query Streaming. The client constructs an initial query using an attached snippet of an audio (1), where the server finds 100 items that is similar to the client’s example (2). The client asks for the genres associated with those 100 items (3), and the server replies with the genres of “rock”, “country”, and “classical” (4). The client then updates the original query with an additional term of “genre is country” (5), and the server reports that there is one file matching the query (6).
ample provided by the user was not of sufficient length or contains noise. The actual result set for the 100 matches are kept in the server.

3. the client asks the server what is the genres contained in the result set using an FRU query.

4. the server replies using FUU: *The genres in the result set are rock, country and classical.*

5. the user sends an additional query term using FRU: *Filter the result set: the genre is country.* The instruction were received by the server, and the server updates the original query to reflect the change (shown in Figure E.0.7).

6. the server finds exactly one match for the given criteria.

The example MQF instantiations of each step in the query is depicted in Appendix E.

The overall communications using MQF, FRU and FUU are compressed using BiM to lower the XML overhead. The result with and without BiM compression are shown in Tables 3.7.5 and 3.7.6. In Tables 3.7.5 and 3.7.6, it was shown that using BiM resulted in significantly lowered overhead (the MQF portion of the XML) compared to without using BiM. On average (for the audio query scenario), the BiM-compressed MQF files are 14% the size of the original uncompressed MQF files.

3.8 MQF and the MPEG Query Format Standard

The MP7QF call for proposal [99] put forth in 2006 generated interest in the research community, and resulted in eight proposals which are received by MPEG during the MPEG meeting in Marrakech, Morocco in January 2007 [112] (the MQF proposal was detailed in [113]). Some of the proposals were published in [114, 115, 116, 117, 118, 119], and the MQF proposal was published in [101]. Discussions and suggestions from the participants including the work described in this chapter took place during the Marrakech MPEG meeting, which leads to further experiments and updated requirements to refine MP7QF that were reflected in five documents:

1. *query expression capabilities* [107] where the RPN approach of MQF was one of the four query methods considered along with three other proposed approaches: using a special query operator description, using XML inheritance method to express a query, and using XQuery as the query language.

2. *client specification of the output format* [108] where the XPath approach of MQF (i.e. the specification of the output format was defined in terms of
## Chapter 3. A New Multimedia Query Format

<table>
<thead>
<tr>
<th>XML file</th>
<th>Original size (bytes)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total size</td>
<td>Binary attachment size (text format)</td>
<td>MQF size</td>
</tr>
<tr>
<td>Initial query with embedded binary</td>
<td>5,181</td>
<td>3,828</td>
<td>1,353</td>
</tr>
<tr>
<td>Initial server reply</td>
<td>375</td>
<td>-</td>
<td>375</td>
</tr>
<tr>
<td>Query refinement</td>
<td>375</td>
<td>-</td>
<td>378</td>
</tr>
<tr>
<td>Final server reply with embedded binary</td>
<td>5,690</td>
<td>3,828</td>
<td>1,862</td>
</tr>
</tbody>
</table>

Table 3.7.5: Audio query scenario XML files, uncompressed.

<table>
<thead>
<tr>
<th>XML file</th>
<th>Compressed size (bytes)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total size</td>
<td>Binary attachment size (binary format)</td>
<td>MQF size</td>
</tr>
<tr>
<td>Initial query with embedded binary</td>
<td>2,835</td>
<td>2,794</td>
<td>41</td>
</tr>
<tr>
<td>Initial server reply</td>
<td>40</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>Query refinement</td>
<td>136</td>
<td>-</td>
<td>136</td>
</tr>
<tr>
<td>Final server reply with embedded binary</td>
<td>3,051</td>
<td>2,794</td>
<td>257</td>
</tr>
</tbody>
</table>

Table 3.7.6: Audio query scenario XML files, BiM compressed.
the structure of MPEG-7 description format) was one of the four methods considered along with three other proposed approaches: using XSLT to define the structure of the result set, using XQuery to define the structure of the result set, and fixed-element approach which specifies the elements that are to be returned by the server.

3. **output query format** [109] where the MPEG-7 subset approach of MQF (i.e. the server returns result using a subset of MPEG-7 descriptions) was one of the five approaches considered along with four other approaches: pure MPEG-7 return format, pure MPEG-7 return format with grouping and item type information, XQuery-specified output format, and a special record type which “flattens” the MPEG-7 description structure (i.e. removing parts of the hierarchical descriptions of MPEG-7).

4. **querying using textual description** [110] where the built-in support for textual query of MQF (i.e. the use of a “freetext” query level) was one of the four methods considered along with three other methods: using EBNF syntax, using XQuery, and using the MPEG-7 description schemes to express textual descriptions.

5. the updated MPEG-7 Query Format requirements from the input of the participants and discussions during the MPEG meeting, which includes a more detailed use case scenario compared to the original 2006 call for proposal [120].

All experiment participants including the present author contributed to the queries to be used during the experiments, which resulted in 96 queries of various scenario (which includes input queries, output queries, query management, error reporting, etc.). All participants were required to formulate example queries using their methods, the results of which were further discussed along with the technical merits of each proposal. During the experiment, MQF consistently achieved low query size and low processing requirements (in terms of memory usage) compared to other proposals. The results of the discussion among the participants in the standardization process was published in [100].

The initial work in the MP7QF standardization process resulted in an ongoing standard which is the more general query format known as the MPEG Query Format (MPQF) [121], and was created from the suggestions, observations, and improvements resulting from the standardization process participants. Similar to MQF, the upcoming MPQF standard decouples the previously MPEG-7 centric focus of the format to a query format that is able to include description schemes other than MPEG-7 to serve as query terms.
CHAPTER 3. A NEW MULTIMEDIA QUERY FORMAT

3.9 Conclusions

This chapter has presented the Multimedia Query Format (MQF) that was designed to serve as a communication format for querying multimedia databases by content. MQF makes it possible to formulate complex query conditions by using any combination of multimedia content in any format with the use of RPN notation and the introduction of the concept of Query Abstraction Levels. Example instantiations of MQF has highlighted its simplicity and flexibility. Its use of XML also allows MQF to work with existing web standards for information interchange such as SOAP and WSDL, and at the same time the abundance of XML tools are immediately available for MQF.

Although MQF provides a protocol for communication between a client looking for a certain multimedia item and a server which contains the multimedia item in question, the MQF specification does not govern how the multimedia data should be stored and structured in the server.

MQF does not aim to replace any internal query language already implemented in any database, but instead extends them (either using MQF alone or MQF combined with FRU and FUU) to provide a consistent external-facing multimedia query interface that could scale from mobile devices to desktop applications using one query format. With the availability of MQF + FRU/FUU, a multimedia database without mobile device capabilities can readily implement a query translator that translates between its internal query languages to MQF and vice versa and thus instantly gain mobile capabilities.

Combining MQF with FRU/FUU allows for targeted browsing of result sets using metadata returned by the server rather than the results themselves. This is particularly suited to querying multimedia databases using mobile devices with a limited or unreliable network connection. Results for typical use case scenarios highlight the advantages of this solution in relation to minimization of the query communication bit rate. For the typical use cases examined in Section 3.7.3, Query Streaming combined with BiM compression leads to a significant reduction in the transmission sizes required to perform multimedia querying.

Furthermore, MQF was submitted to MPEG as a proposal for the MPEG-7 Query Format (MP7QF) standardization process. Tests involving different use case scenario provided by MPEG show that MQF conforms to the requirements for a modern query format as defined by MPEG to perform multimedia search.
Chapter 4

Description of Human Motion

4.1 Introduction

Chapter 3 has described the Multimedia Query Format (MQF), which is a new query format designed to allow query communication between client and multimedia databases. MQF was designed to communicate query terms that are not specific to any particular multimedia description scheme, as long as XML is used for the description format.

In the overall goal of this thesis depicted in Figure 4.0.1, multimedia search
using human motion requires three separate solutions: the query communication format (of which MQF was designed for); a method to describe human motion that is searchable; and a method to extract such descriptions from raw data. Therefore in the context of the thesis, this chapter will present a new, detailed and searchable human motion description format.

Currently, technologies for searching a specific human motion in multimedia is lacking a semantic, detailed description of human motion that facilitates search. For example, the currently available tools can be roughly categorized as text-based search and query-by-example:

1. **text-based search**: Text-based search relies on the use of keywords to describe the desired media. For example, searching for a walking motion using the keyword of “walking” or “walk”. This has the advantage of relative simplicity, i.e. the user needs to only type “walking” into a search box, and the server would return all media with the keyword “walking” associated with it and leave the work of sorting through the result set to find the desired media to the user themselves. However, the multimedia database would have to be tagged correctly (e.g. a video of John walking cannot be tagged as “John” without specifying the motion he is performing in the video, and the clip could also contain many other motions besides walking), otherwise the result set would contain many irrelevant results. Text-based search cannot adequately describe human motion as remarked by Laban [14], where he described in detail that describing a person walking using text in any reasonable amount of detail can quickly become an exercise in complexity. For example, describing a person walking, one might need to specify the number of steps, the timing of the steps, the length of the steps, etc.

2. **query-by-example**: Using an example of a media to serve as the query term. For example, searching for walking motions using a walking motion itself as the query. This has the advantage of a more detailed query “term” compared to a text-based search. However, the disadvantage is the need for the user to obtain the example itself, which could be difficult if the user requires a very specific motion (such as a contemporary dance motion, which may not be easily obtainable). Also, since there is no standard on how human motion should be described, the one example the user provides could be interpreted differently from server to server. For example, one server could interpret the motion as “dancing” in general terms (which would return all dancing motions), the other could interpret it as a series of parameters of how the limb is moving (which would return similar limb movements but does not differentiate between a slow motion and a fast motion), yet another server could interpret it as a
series of features in time (which would return motions with similar timing but
not necessarily the same motion), etc.

The two aforementioned methods cater to two different extremes of motion
search: text search is too general (thus requiring a detailed description in the mul-
timedia database), and query-by-example is too detailed (thus requiring a detailed
query term). It is therefore desirable to have something in-between those two ex-
tremes.

Laban’s solution to complex textual description of human motion is a special-
ized notation termed “Labanotation” [14, 78]. With roots in dance, Labanotation
describes the aforementioned walking motion in a precise manner required for dances
using as few as four symbols. It is a testament to Laban’s observation for the need to
describe precise human motion that his notation is still being used today, essentially
unchanged as to his original design in the 1920’s. In fact, there are projects aiming
to employ Labanotation to computer interfaces [85], highlighting its flexibility.

One immediate shortcoming of Labanotation for multimedia search use is its root
in dance. Dance utilizes timing derived from music (e.g. the movements are timed
according to “beats” and “measures” commonplace in musical notations), and Laban
designed his notation to be used with those musical notations in mind. Attempts
to bring Labanotation to the computer age such as MovementXML [84] are still
burdened with this limitation. The underlying movement symbols of Laban are
quite universal, however, and inspired the work in this chapter.

This chapter provides a method to describe human motion in detail to facilitate
human motion search, and is arranged as follows: Section 4.2 provides the motiva-
tions for the need for a detailed human motion description, Section 4.3 provides a
detailed design description of Human Motion Markup Language (HMML), Section
4.4 provides examples of how HMML can be used for search and description of hu-
man motion, and Section 4.5 shows how HMML can be encapsulated in MQF in a
query involving multimedia description and human motion.

4.2 Motivations

The motivations to develop Human Motion Markup Language are:

- a method to describe human motion in detail without resorting to
  complex textual descriptions.

As noted by Laban [14], even describing a natural motion such as walking
can become excessively complex if the desired motion is precisely defined. For
example: “walking two steps” can also be described as “walking two steps of
regular size” or “walking two steps of regular size using regular timing”. These
textual descriptions lack the definition of “regular size” and “regular timing”, which can lead to imprecise queries and descriptions. Therefore, a human motion description should adopt a more symbolic representation of human motion instead of using words to describe a motion (as concluded by Laban in [14]).

- **query-by-example without having any examples.**
  Query-by-example allows a more precise query term to be formed compared to textual descriptions of a motion by providing an example motion to serve as a query term. However, if the desired motion is unconventional (e.g. a dramatic walk) or very specialized (e.g. a gymnastic routine or a contemporary dance), the example motion may not be easily obtainable. A human motion description solution would need to allow the user to create a motion description that is sufficiently detailed to facilitate search.

- **a human readable motion description.**
  To enable query-by-example without an example, the motion description language would have to be human-readable and relatively straightforward to create. This also enables the motion description language to serve as “motion-level metadata” (i.e. semantic-level metadata) to complement “high-level metadata” (e.g. Dublin Core) and “low-level metadata” (e.g. MPEG-7) that could be attached to the motion recording to provide a more complete description of a media.

- **a multimedia-oriented Labanotation-inspired description of human motion.**
  Labanotation was originally designed for dances accompanied by music, hence Labanotation’s timing information depends heavily upon musical timings such as “beats” and “measures”, resulting in timing elements that cannot be easily decoupled from the music it is based on, due to the relativity of the definition of a “beat”, which could differ from motion to motion. On the other hand, computer-based video and motion capture base their timing information on more precise “frames per second” measurements. A multimedia-oriented notation should use the more precise timing definition as the basis for the motion timing information to enable the synchronization of the motion description to a video recording.

Although the movements of the human body are defined in terms of the three planes (sagittal, coronal, and transverse), Labanotation combines the description of sagittal and coronal plane for readability purposes, and there are only
three possible movements in the transverse plane (high, middle, and low). La-}
banotation is not meant to be a recording (in multimedia terms) of a motion, and reading it requires human interpretation that is subjective, which was designed by Laban into the notation to avoid excessive cluttering of the notation [14]. A computerized motion description should not leave vague interpretation and should act as a faithful representation of the actual motion performed.

- a server-to-server communication language for motion search interoperability.

The motion description language can also serve as a baseline language to describe human motion in a similar fashion to the interoperability of office applications afforded by the Open Document Format (ODF) [122]. Given a good text-to-motion-description automatic conversion, a user might not necessarily see the actual motion description used in the query, but the common motion description format facilitates communication between disparate servers.

4.3 Human Motion Markup Language (HMML)

The goal of the Human Motion Markup Language (HMML) is to provide a baseline motion description language to facilitate multimedia search containing human motion that satisfies the motivations put forth in Section 4.2. The language is based on the ideas put forward by Laban for human motion notation (i.e. Labanotation) with extensions to facilitate computer processing and search capabilities.

HMML also shares a similar spirit to the Hypertext Markup Language (HTML) [37], in that HMML intends to be a platform-independent method to describe content using syntax based on Standard Generalized Markup Language (SGML) [27] (of which Extensible Markup Language – XML [26] is also based on).

XML-based representation of Labanotation exists in the form of MovementXML [84]. However, its use of musical timing information which is inherited from Labanotation limits its flexibility in describing motion in multimedia applications (due to the non-standard definition of its timing information). MovementXML was also not designed with search in mind, but as a pure XML representation of the written (i.e. symbol-based) Labanotation.

4.3.1 Requirements

To achieve the motivations described in Section 4.2, a searchable human motion description format would have to meet the following requirements:
1. **limb based**: To describe human motion accurately, the movement of individual major limbs such as the arms and the legs has to be described as in Labanotation [78]. Labanotation is a limb-based human motion description, proving the flexibility and accuracy of a limb-based approach. In terms of search, a search for a hand waving motion would result in a standing and waving motion and walking and waving motion, in effect expanding or restricting the detail level of the search according to the wishes of the user.

2. **anatomical plane-based**: To describe a motion independent of the direction that the body is facing, the anatomical planes of the human body provides a fixed frame of reference relative to the body itself. This is an extension to the Labanotation symbols, but extensions to cover a mode detailed movements in the transverse plane (of which Labanotation only defined three possible levels of high, low, and middle [78]) and separate description of the sagittal and coronal planes (of which Labanotation combined the two planes into one [78]) are desired to provide a more detailed, searchable description.

3. **non-reconstruction**: Since the description would describe general, temporal movements of the body, reconstruction for motion playback purposes is not needed since the goal of the motion description is not to replicate functionalities already available in e.g. motion capture, but to describe the content of a motion in a searchable manner. In contrast, motion capture was designed to faithfully record a motion and was designed to reconstruct the exact motion recorded without any description of the motion being performed (e.g. the BVH motion capture format discussed in Chapter 2). This requirement would also prevent the description scheme to be too complicated and too detailed to search. However, the subjectivity of a reader of Labanotation (in which the reader would have to infer the most “natural” path for limb movements for each symbol) is not desired since it creates ambiguity.

4. **temporal**: It needs to be able to describe human motion temporally to provide a detailed description that can be used to search for a specific moment in time of a motion, e.g. it has to be able to search for a “running then jumping” motion, with a clear separation between the running and the jumping portions of the motion.

5. **Semantic**: It needs to be able to semantically identify a motion with at least the same amount of detail that Labanotation description of a motion, to facilitate searching human motion using a description of its limb movements. For example, in a walking motion, there are moments when the leg swings forward and the body weight is transferred to that particular leg. The motion
description format has to be able to describe a motion in a fine-grained manner (such as the moment when the leg swings forward), and also in a general manner (e.g. this is a walking motion).

4.3.2 The HMML Schema Design

Taking into account all the outlined requirements in Section 4.3.1 and the insights of Laban [14] (and the XML counterpart of Labanotation – MovementXML [84]), the Human Motion Markup Language (HMML) format has been implemented as an XML Schema. XML was chosen as it provides maximum compatibility between existing popular description formats – Dublin Core can be described using XML, and MPEG-7 is entirely XML-based. The core elements of the HMML schema is as follows):

- the element <motion> that encapsulates the motion description format, with an optional “fps” attribute that specifies the number of recorded frames per second in the annotated motion.

- the elements <sagittal>, <coronal>, and <transverse> that encapsulate descriptions in their respective axis:
  - the sagittal plane divides the body between left and right.
  - the coronal plane divides the body between front and back portions.
  - the transverse plane divides the body between upper and lower portions.

- the elements <leg>, <arm>, <torso>, and <head> that describe the movements of the limbs.

The definition of the elements in the XML Schema are described in Sections 4.3.3 to 4.3.6. In HMML, the motion of a limb is described in three planes simultaneously (sagittal, coronal, and transverse), where each limb has three directional descriptors with the restriction that only one direction is to be specified for a movement (since a limb cannot move in two directions in the same plane simultaneously). Using a plane-based description format, this enables HMML to describe motion in three dimensions, even for a 2D video. For example, the arms in the sagittal plane can move in forward, backward, and center directions. In the coronal plane in right, left, and center directions, and in the transverse plane in high, low, and level directions. Exceptions are the “center” direction that only applies to the sagittal and coronal plane, and the “none” direction that applies to all three planes (which signifies that the limb in question is not moving).
By using the same method to describe all three planes, HMML provides the same level of detail for all three planes, an extension of Labanotation that does not provide the same level of detail for the transverse plane compared to the sagittal and coronal planes. Also, the sagittal and coronal planes are described separately, where in Labanotation both planes are combined for readability reasons.

4.3.3 The root <hmml> Element

The root element of HMML (illustrated in Figure 4.3.1) allows five different sub-elements:

- the <sagittal>, <coronal>, and <transverse> elements (described in Section 4.3.4).
- the <motion> element (described in Section 4.3.5).
- the <repeat> element (described in Section 4.3.6).

Additionally, the root element also defines two attributes:

- “fps”, to specify the frame-per-second of the motion being described.
- “version”, to specify the version of the HMML in use. For this thesis, the “version” attribute is set to “1.0”.

4.3.4 The <sagittal>, <coronal> and <transverse> elements

The <sagittal>, <coronal>, and <transverse> elements form the building blocks of the motion description, where each element describes the limb movements in their specific plane. Figure 4.3.2(a) depicts the schema definition of only the <sagittal> element for brevity, since the <coronal> and <transverse> elements are similarly defined.

Each planar description (sagittal, coronal, and transverse) contains the actual limb movement description. Figure 4.3.2(b) illustrates the <leg> element only for brevity, since the <arm>, <torso>, and <head> elements are similarly defined.

The limb description elements of <leg>, <arm>, <torso>, and <head> specifies attributes that describe how the limb moves with respect to its plane of movement:

- “side” defines which side of the limb is being described, with possible values of “left” and “right”.

<leg side="left"/> denotes that the limb being described is the left leg.
Figure 4.3.1: Illustration of the root element of HMML.
Figure 4.3.2: The primary building blocks of the HMML motion description: (a) the planar element (only the <sagittal> element is shown for brevity) which contains the limb movement description elements shown in (b) where only the <leg> element is shown for brevity. For both elements, the other planar elements (<coronal>, <transverse>), and the other limb elements (<arm>, <torso>, <head>) are defined similarly.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

Figure 4.3.3: An example HMML instantiation describing walking forward four steps starting with the left leg, described in the sagittal plane.

- **“dir”** defines what direction the limb in question is moving toward. Possible values vary according to the plane of movement, i.e. “forward”, “backward” and “none” for the sagittal plane; “left”, “right” and “none” for the coronal plane; and “up”, “down”, and “level” for the transverse plane.

  \[<\text{leg side=“left” dir=“forward” start=“1” end=“60”}/>\]

  \[<\text{leg side=“right” dir=“forward” start=“60” end=“120”}/>\]

  \[<\text{leg side=“left” dir=“forward” start=“120” end=“180”}/>\]

  \[<\text{leg side=“right” dir=“forward” start=“180” end=“240”}/>\]

- **“start”** and **“end”** specifies the starting frame and ending frame of the movement.

  \[<\text{leg side=“left” dir=“forward” start=“1” end=“60”}/>\] denotes that the left leg is moving forward from frame 1 to frame 60.

- **“duration”** specifies the duration of the movement. This attribute can be used by itself or in conjunction with the **“start”** attribute.

  - \[<\text{leg side=“left” dir=“forward” duration=“60”}/>\] denotes that the left leg is moving forward for 60 frames, or
  
  - \[<\text{leg side=“left” dir=“forward” start=“1” duration=“60”}/>\]
    
    denotes that the left leg is moving forward for 60 frames, counting from frame number 1.

In combination, an example instantiation describing a four-steps walking motion is shown in Figure 4.3.3.

In a description, HMML does not require a rigid frame number specification to be present (i.e. the **“start”** and **“end”** attributes are not required to create a valid motion description). The reason for the inclusion of the **“start”** and **“end”** information is to facilitate a detailed motion description; for example, it is not possible to differentiate between a fast walk and a slow walk without knowing the duration of the leg movements. However, by omitting the motion duration information, both motions are still recognized as a walking motion.
The main purpose of the <motion> element is to provide the ability to group logical and/or simultaneously occurring motions together in a “container”. For example, a user may want to separate a compound motion of walking and running into its logical parts of “walk” and “run”. The <motion> element may contain:

- three sub-elements that describe movements in a specific plane as previously described in Section 4.3.4: the <sagittal>, <coronal>, and <transverse> elements.
- more <motion> elements (as a recursive definition).
- <repeat> elements (described in Section 4.3.6).

The <motion> element can be instantiated with the optional attribute of:

- “id”, to uniquely identify a motion grouping. There is no restriction on the format of the “id” attribute, for example, <motion id="walk">, <motion
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

id="123">, or <motion id="23EA77"> are all valid.

- "start" and "end", to specify the starting and ending frame number of a grouped motion similar to the limb movement description in Section 4.3.4

- "duration", to specify the duration of the grouped motion similar to the limb movement description in Section 4.3.4.

Grouping logical motions

The <motion> element defines the attribute of "id", which assigns a name to the grouped motion. An example instantiation is shown in Figure 4.3.5. This example shows two <motion> element encapsulating the walk forward four steps motion previously shown in Figure 4.3.3 and the run forward four steps motion. Note that for illustrative purposes, the only difference between walking and running motions are the duration over which the legs move forward. In the example shown in Figure 4.3.3, the duration of leg movements in walking are 60 frames, and for run motion are 40 frames (which denotes faster leg forward movement). In order to logically group the walking vs. the running motions, each motion are put into their own separate <motion> elements with a descriptive name (i.e. <motion id="walk"> and <motion id="run"> for the example in Figure 4.3.4).

Grouping simultaneously occurring motions

If limb movements are described without using the "start" and "end" attributes (e.g. the limb movement is only described using the "duration" attribute as described in Section 4.3.4), there is no information regarding the order of occurrence of the limb movements. An example of this possible scenario is shown in Figure 4.3.6. In Figure 4.3.6, there are four leg movements; namely, two leg forward movements of the left leg and two leg forward movements of the right leg. While the intention of the HMML instantiation in Figure 4.3.6 is to describe a walking forward motion with four steps starting with the left leg, the absence of absolute time element could result in ambiguous interpretation; i.e., the HMML can be read as a person having four legs all moving forward at the same time.

To overcome this possible ambiguous interpretation, motion grouping using the <motion> element is used as shown in Figure 4.3.7. This possible ambiguous interpretation is used for querying motion and not for describing an existing motion, the example of which will be shown in Section 4.4, and forms the basis of the <repeat> element in the following Section 4.3.6.
Figure 4.3.5: An example HMML instantiation of a logically grouped motion. In this case, walk and run.

Figure 4.3.6: Ambiguous order of leg movements without the use of the “start” and “end” timing attributes, where the four leg movement descriptions have no obvious order.
Figure 4.3.7: Removing ambiguous interpretation of movement occurrence order using motion grouping.
High-level description of a motion

While typically video sharing sites assign tags to the video as a whole, Davis et al. experimented with user-supplied temporal video tagging in [5] that assigns tags to sections of a video. For example, in a video, John Doe was seen running (temporally tagged as “run”) and jumping (temporally tagged as “jump”) while the clip as a whole could be tagged as “John Doe”, “run and jump”. This temporal tagging approach thus allows for a more detailed description of a video.

Similarly, the `<motion>` element can be instantiated without any limb movement description to achieve temporal, high-level description of a motion. This feature is useful in case a high-level motion description is desired (e.g. “walk” or “run” instead of describing limb movements), or a manual method is used to annotate an existing motion (such as in videos as in [5]). An example of this high-level description is shown in Figure 4.3.8.

The example in Figure 4.3.8 describes a general description of two distinct smaller motions: walking (from frame 1 to 240) followed by running (from frame 240 to 400) inside a long motion, where a more detailed description involving detailed individual limb movements was previously shown in Figure 4.3.5. This feature allows flexibility in manual annotation of a motion, in that the user is not forced to be overly detailed in describing all the limb movements in a motion if it is not required.

The high-level description of the `<motion>` element also allows searches based on the “id” attribute. For example, textual searches with keywords of “walk and run” and searches with a keyword of only “walk” would match the description given in Figure 4.3.8, allowing HMML to interoperate with existing text-based search methods.

Figure 4.3.8: High-level description of a motion using only `<motion>` elements to describe a walking motion followed by a running motion. This example is a more general representation of the motion described in Figure 4.3.5.
### 4.3.6 The `<repeat>` element

The `<repeat>` element definition is similar to the `<motion>` element described in Section 4.3.5 with an important distinction: the `<repeat>` element allows a grouped motion to be repeated a specified number of times. This element is functionally similar to the Labanotation “repeat” symbol, where a certain motion is to be repeated (this concept is written in musical notation as the “repeat” sign).

The `<repeat>` element may contain:

- three sub-elements that describe movements in a specific plane as previously described in Section 4.3.4: the `<sagittal>`, `<coronal>`, and `<transverse>` elements.
- more `<motion>` elements (described in Section 4.3.5).
- more `<repeat>` elements (as a recursive definition).

The `<repeat>` element can be instantiated with the optional attributes of:

- “id”, to uniquely identify a `<repeat>` grouping. This identification tag is similar to the `<motion>` element “id” tag described in Section 4.3.5.
Figure 4.3.10: An example HMML instantiation of a repeated motion. The example shows two steps forward repeated twice, resulting in walking four steps forward. Each of the leg-forward motion would need to be encapsulated inside a <motion> element to remove ambiguity in movement order as described in Section 4.3.5.
4.4 Semantic Motion Query Using HMML

As described in Section 4.2, one of the primary motivations driving the creation of HMML is semantic querying of human motion. As discussed in Sections 4.1 and 4.2, the lack of a standardized language describing human motion is one of the hindrances to achieve this goal. This Section thus provides a method to perform semantic queries involving human motion by leveraging HMML, either alone or with other multimedia description techniques such as MPEG-7 and Dublin Core by forming a “filter”.

4.4.1 The HMML description of a walking Motion

To illustrate HMML, this Section utilized the diagram illustrated in Figure 4.4.1 to represent the XML instantiation of HMML. Figure 4.4.1 illustrates a description of a walking forward four steps motion: the left leg stepping forward at the same time as the right arm moving forward and the left arm moving backward (one step of the left leg in a walking motion), where the movements are observed from the sagittal plane. The next set of movements are the right leg stepping forward along with the right arm moving backward and the left arm moving forward (one step of the right leg in a walking motion). The two motions (left leg stepping forward and right leg stepping forward) are repeated once, completing the description of a walking forward four steps motion. The XML instantiation of the illustration of Figure 4.4.1 is depicted in Figure 4.4.2.

For the purposes of this Section, the HMML document illustrated in Figure 4.4.1 (and instantiated in XML shown in Figure 4.4.2) is assumed to exist in the server, and is the description to be searched. Section 4.4.2 will discuss how a temporal query using this description will be formed.

4.4.2 Using HMML description fragments to form a temporal query

Temporal and/or semantic queries are one of the requirements for the Multimedia Query Format (MQF in Chapter 3) and MPEG-7 Query Format. While HMML is a description format, fragments of HMML can be employed to perform temporal queries. Some properties of the HMML design such as describing motion per-plane and per-limb can be exploited to form queries by using HMML itself to serve as the query term.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

Figure 4.4.1: An illustration diagram of HMML describing leg and arm movements temporally in the sagittal plane.

Limb-based temporal query

An illustration of using a HMML fragment-to-HMML description query is shown in Figure 4.4.3 by constructing parts of a walking motion, such as walking four steps in a forward direction starting with the left leg as viewed from the sagittal plane (i.e. from the side). The query portion of the illustration in Figure 4.4.3 (with example XML fragment shown in Figure 4.4.4) forms a subset of the walking motion description shown in Figure 4.4.1 and the XML in Figure 4.4.2. To perform a temporal search for a motion, the server would therefore search for a description that is the superset of the incoming query.

By matching a subset of an existing description, it is possible to perform a search using only a specific body plane, since the body planes divides the body in a constant manner no matter the direction that the body is facing. For example, when searching a walking forward motion, the coronal plane movement is irrelevant. Similarly, when searching for a sidestepping motion, the movement of the sagittal plane is, in turn, irrelevant.

Additionally, it is also possible to perform a search using only a specific limb. For example, if the user is only interested in the movement of the legs and would like to search for any motion involving walking without regard to what the arms are doing. This is also illustrated in the example shown in Figures 4.4.3 and 4.4.4, where the user formed the query using leg movements only. This example query would match walking normally with swinging arms, walking while waving, walking with the arms not moving (i.e. military-style walk), etc. This query would also match walking with different speeds (fast walk, slow walk, etc.) since the query only specifies the desired movements of a specific limb and not the detailed timing.
Figure 4.4.2: HMML instantiation in XML for the diagram shown in Figure 4.5.1. This document is assumed to exist in the server.
Figure 4.4.3: An illustration of HMML temporal matching example by matching the incoming query to a subset of an existing description. The incoming query (labeled as “Query” in the diagram) matches part of the description stored in the database (labeled as “In Database” in the diagram).

```xml
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:uow:hmml:schema:2012 HMML.xsd"
  fps="120">
  <sagittal>
    <leg side="left" dir="forward" start="1" />
    <leg side="right" dir="forward" start="60" />
    <leg side="left" dir="forward" start="120" />
    <leg side="right" dir="forward" start="180" />
  </sagittal>
</ns:hmml>
```

Figure 4.4.4: Example XML of an HMML fragment forming a temporal query (labeled as “Query” in Figure 4.4.3).
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

information associated with the movement.

Searching for a repeating motion

Another feature of HMML is the possibility to search for any number of repetitions of movement as detailed in Section 4.3.2 by using the `<repeat>` element to form a query. For example, the user can specify two steps of a walking motion (*left leg forward – right leg forward*) and encapsulate the description inside a `<repeat times="2"/>` element. In other words, the user creates a group of movements that is to be repeated twice, forming the query. An illustration of this query term is shown in Figure 4.4.5.

Note that the example instantiation of the query shown in Figure 4.4.6 encapsulates the leg movement description inside a `<motion>` element to prevent ambiguity in movement order as previously described in Section 4.3.5.

Another scenario involving repeating motion grouping is the use of the `<repeat>` element’s “*times=unbounded*” attribute. For example, if the user wanted to search for a walking motion by describing the movement and timing of the legs, but doesn’t care about the number of steps taken. The `<repeat>` element described in Section 4.3.6 can be used for this purpose.

Semantic motion search using a textual query

Traditional keyword search can be achieved by leveraging the “*id*” attribute in the `<motion>` element. Assuming that the “*id*” attribute describes (“tags”) the semantic content of the motion in the server, these tags can serve as a semantic content identifier. For example, the XML instantiation shown in Figure 4.4.2 contains the element `<motion id="walk">`. By searching using the keyword “walk”, the whole element is matched and returned.

A more detailed example is illustrated in Figure 4.4.7, where a motion is tagged as consisting of four types of motions: walk; run; jump; and turn. Note that the motions are all tagged in a more “traditional” manner, that is, the exact movements of the limbs are not specified. Keyword search using any of the tags in Figure 4.4.7 will return the whole motion, optionally with the range of frames (that is specified in each `<motion>` element) when that sub-motion is occurring.

Using HMML in conjunction with Dublin Core and MPEG-7 for search result filtering

One of the goal of HMML (as described in Section 4.3) is the use of HMML as a motion description format to be used in conjunction with other, more established multimedia description format such as Dublin Core or MPEG-7 to provide a more
Figure 4.4.5: An illustration of matching repeating group using the <repeat> element to match walk four steps by using walking two steps repeated twice. An example XML instantiation of the Query portion is shown in Figure 4.4.6.
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="1.0" fps="120">

<repeat times="2">
    <motion>
        <sagittal>
            <leg side="left" dir="forward" duration="60"/>
        </sagittal>
    </motion>

    <motion>
        <sagittal>
            <leg side="right" dir="forward" duration="60"/>
        </sagittal>
    </motion>
</repeat>
</ns:hmml>

Figure 4.4.6: The XML instantiation of the query portion of the diagram shown in Figure 4.4.5. Each of the leg-forward motion would need to be encapsulated inside a <motion> element to remove ambiguity in movement order as described in Section 4.3.5.

<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    <motion id="walk" start="1" end="240"/>
    <motion id="run" start="240" end="360"/>
    <motion id="jump" start="360" end="380"/>

</ns:hmml>

Figure 4.4.7: Example XML instantiation of a motion tagged by its semantic content using textual tags. In this case, “walk”, “run”, and “jump”. Each tagged content specifies the starting and ending frame of the sub-motion. Textual search using keywords of “walk”, “run”, and/or “jump” will return this document.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

John Doe

... running and jumping into ...

... a lake

Figure 4.4.8: An illustration of using HMML in conjunction with Dublin Core and MPEG-7 to perform a detailed search of a video using the query terms as an increasingly strict filtering process.

complete description of the media. For example, if a user desires to search of a video containing “John Doe running and then jumping into a lake”, a query using HMML in conjunction with standardized multimedia description formats can be thought of as a series of increasingly strict search filters.

Figure 4.4.8 illustrates the flow of the filtering process involved in a detailed temporal search using human motion. The query terms (“John Doe running and jumping into a lake”) can be separated according to their Subject-Verb-Object construct, with the subject described using Dublin Core (John Doe), the verb using HMML (…running and jumping into…), and the object using MPEG-7 (…a lake). Working together, the Subject-Verb-Object construct provides an increasingly strict filtering criteria for the media to be searched: Assuming that John Doe is searching for his own video that he uploaded to YouTube (100 of them in this scenario), he is searching for a video of himself running and jumping (there are 10 videos that matches the description), and only one of him actually jumping into a lake.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

The example in Figure 4.4.8 illustrates HMML working together with existing multimedia description formats, the combination of which can provide a detailed temporal and semantic search using human motion.

4.5 HMML Query Using MQF

One of the goal of the work presented in Chapter 3 is the creation of a multimedia query format that is capable of using a combination of metadata to form a query, where querying multimedia by its human motion description was one of the design goal of MQF. By using HMML in conjunction with MQF, complex query involving different level of details (exact, example, semantic, and freetext as defined by MQF in Chapter 3) can be constructed with HMML describing the semantic (i.e. human motion) aspect of the query, similar to the diagram shown in Figure 4.4.8. Using MQF, the server can describe the resulting matches from different aspects using a video link, MPEG-7 descriptions, Dublin Core descriptions, HMML, etc. This enabled the user to have as much information about the media as possible during searches. This Section provides example usage scenarios involving single-server query and multi-server query.

4.5.1 Single Server Query Scenario

Since MQF is capable of mixing different description schemes inside a single query or reply, a complex query that describes multiple aspects of the desired match can be performed. For example, the diagram in Figure 4.5.1 depicts the scenario of: user would like to search for a video where “A person is running and jumping into a lake”, but does not remember other detail about the video in question. In this example, the semantic term is “running and jumping”, and it is described using HMML. The user constructs the query using two parts: the HMML description of a person running and jumping, and a freetext description of the query. Both terms are encapsulated using MQF. An illustration of this scenario is shown in Figure 4.5.2, with the XML instantiation of the Query and Reply parts of Figure 4.5.2 are shown in Figure 4.5.2 and 4.5.4, respectively.

Figure 4.5.1 consists of two general parts: the Query part; and the Reply part.

The Query part (with example XML instantiation in Figure 4.5.2 and 4.5.3) is communicating a query of “search for a video containing the motion running and jumping into a lake” by using two query terms: the semantic-level HMML term of “run” and “jump”, and the freetext-level term of “running and jumping into a lake”.

Figure 4.5.2 illustrates a simple semantic query by using HMML’s <motion> using the “id” attribute that describes the motion being performed, while Figure
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

Figure 4.5.1: An example scenario of a multimedia query using HMML transported in MQF.
4.5.2 Multi Server Query Scenario

By leveraging the capabilities of MQF to perform multiple searches simultaneously using a meta-search (i.e., aggregation) server, Figure 4.5.5 shows an example of a meta-search scenario. The client connects to a meta-search server that retains a list of human motion enhanced search capable multimedia databases. When a query is received by the meta-search server, it is relayed to all server in its database, and
Figure 4.5.3: MQF-encapsulated HMML search representing the query portion of the diagram shown in Figure 4.5.1.
Figure 4.5.4: MQF-encapsulated HMML query reply (using the example shown in Figure 4.5.1). Metadata returned included Dublin Core and MPEG-7.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

4.5.5: Querying Human Motion Description using MQF: multiple server query scenario.

the replies are aggregated for the client. An illustration of this scenario is shown in Figure 4.5.5.

The user query scenario in Figure 4.5.5 is similar to the example in Figure 4.5.1 (searching for a movie where “a person is running and jumping”). However, in this example, the multimedia databases contain different multimedia description formats. Multimedia Database 1 (marked as “Server 1 reply” in Figure 4.5.6) is using HMML, MPEG-7, and Dublin Core, and Multimedia Database 2 (marked as “Server 2 reply” in Figure 4.5.6) is using only HMML to describe their contents. The meta-search server aggregates the results of both databases with different level of description details while noting clearly each server’s results. By using MQF encapsulation, the disparate multimedia description schemes can be returned as a single XML document to the client. The XML representation of the aggregated reply is shown in Figure 4.5.6.

4.6 HMML Size Comparison Experiments

In this Section, experiments are performed to measure the size of an HMML annotation compared to a motion recording. For the purpose of this test, the motion
Figure 4.5.6: XML representation of aggregated MQF encapsulated query reply from two multimedia databases.
capture recording in the BVH format was used, due to the specifications of both motion capture and HMML that describes movements in three-dimensions. Although HMML was primarily an interchange format for motion annotation that could also annotate motions recorded in two-dimensions such as video, the problem of limb occlusion (where one limb is occluded by another object e.g. the torso) means that the HMML annotation formed from a two-dimensional source potentially does not cover all the movements which are occluded in the recording, resulting in an artificially smaller HMML description. Hence, in order to fully annotate the recorded motion, 3D recordings of motion are used as the annotation source in this experiment.

For this experiment, the motion used are stored in the BVH motion capture format. The motion files are acquired from the HDMcuts database [123], and the files to be transcribed are arranged into four types of motion as shown in Table 4.6.1: hopping forward three times (12 files); jumping jack repeated three times (13 files); walking forward four steps (16 files); and walking backward four steps (15 files). The total number of files used in this experiment is 56 files.

### 4.6.1 Methodology

Each of the files in the HDMcuts database are annotated according to the HMML limb movement specifications by manually observing the movements of the limbs in the animated playback of the motion recordings, and the resulting annotation (i.e. the resulting HMML files) are measured in terms of the total HMML elements used, and the total XML instantiation file size (in Bytes) for each individual files. Further, the HMML files are compressed using two compression methods: gzip and Binary MPEG Format for XML (BiM) [43], which is an MPEG standard that was specifically designed to compress XML documents. The number of elements involved in the HMML annotation shows the complexity of the movements, and the size of the compressed and uncompressed HMML files shows the total bytes that is required to describe the corresponding recorded motion. For comparison, the size of the original BVH recording of the motion is measured in an uncompressed and compressed (using gzip [124]) state. Gzip compression was used in this experiment due to its popularity and its notable use as one of the HTTP compression method.

<table>
<thead>
<tr>
<th>Motion type</th>
<th>Number of files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop forward</td>
<td>12</td>
</tr>
<tr>
<td>Jumping jack</td>
<td>13</td>
</tr>
<tr>
<td>Walk forward</td>
<td>16</td>
</tr>
<tr>
<td>Walk backward</td>
<td>15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>56</strong></td>
</tr>
</tbody>
</table>

Table 4.6.1: The number of files and the motion types used in the size testing.
CHAPTER 4. DESCRIPTION OF HUMAN MOTION

Figure 4.6.1: A typical example of HMML annotation of a walking forward four steps motion.

Table 4.6.2: Average uncompressed and compressed size of the motion files in BVH format used in the HMML annotation testing, grouped by motion types.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Uncompressed (bytes)</th>
<th>Gzip compressed (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop forward</td>
<td>221,889.92</td>
<td>80,800.75</td>
</tr>
<tr>
<td>Jumping jack</td>
<td>328,346.00</td>
<td>119,967.77</td>
</tr>
<tr>
<td>Walk forward</td>
<td>237,066.63</td>
<td>85,321.00</td>
</tr>
<tr>
<td>Walk backward</td>
<td>290,087.87</td>
<td>103,719.20</td>
</tr>
<tr>
<td>Average</td>
<td>269,347.60</td>
<td>97,452.18</td>
</tr>
</tbody>
</table>

Table 4.6.2 shows the average file size (original uncompressed BVH and gzip compressed BVH, in bytes) of the original BVH recordings of four motion classes: hop forward; jumping jack; walk forward; and walk backward. Table 4.6.2 shows that compressing the original BVH files using gzip result in files that are 36.2% the size of the uncompressed BVH, with relatively consistent gzip compression ratio between complex motion such as jumping jack and relatively simpler motion such as walking (36.4% for hop forward, 36.5% for jumping jack, 36.0% for walk forward, and 35.8% for walk backward). These consistent compression ratios indicate that transmitting a human motion query using BVH as the query term yields a predictable bandwidth usage (with or without compression); however, motion complexity is not a deter-
## CHAPTER 4. DESCRIPTION OF HUMAN MOTION

### Table 4.6.3: Average HMML element count and HMML file sizes achieved by annotating the files shown in Table 4.6.2. On average across four motion classes and 56 motion files, the HMML size (in Bytes) are 99.6% smaller compared to the size of the original BVH files.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Element count</th>
<th>Uncompressed (bytes)</th>
<th>Gzip compressed (bytes)</th>
<th>BiM compressed (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop forward</td>
<td>22.75</td>
<td>1,342.25</td>
<td>464.00</td>
<td>122.83</td>
</tr>
<tr>
<td>Jumping jack</td>
<td>42.15</td>
<td>2,385.77</td>
<td>613.15</td>
<td>216.31</td>
</tr>
<tr>
<td>Walk forward</td>
<td>12.44</td>
<td>840.25</td>
<td>382.31</td>
<td>81.00</td>
</tr>
<tr>
<td>Walk backward</td>
<td>10.67</td>
<td>756.07</td>
<td>383.67</td>
<td>85.53</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>22.00</strong></td>
<td><strong>1,331.08</strong></td>
<td><strong>460.78</strong></td>
<td><strong>126.42</strong></td>
</tr>
</tbody>
</table>

### Table 4.6.4: Size comparisons of gzip-compressed HMML and BiM compressed HMML against uncompressed HMML, and uncompressed HMML and BiM-compressed HMML against original uncompressed BVH sizes.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Vs. uncompressed HMML</th>
<th>Vs. uncompressed BVH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gzipped HMML</td>
<td>BiM</td>
</tr>
<tr>
<td>Hop</td>
<td>34.57%</td>
<td>9.15%</td>
</tr>
<tr>
<td>Jumping jack</td>
<td>25.70%</td>
<td>9.07%</td>
</tr>
<tr>
<td>Walk forward</td>
<td>45.50%</td>
<td>9.64%</td>
</tr>
<tr>
<td>Walk backward</td>
<td>50.75%</td>
<td>11.31%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>39.13%</strong></td>
<td><strong>9.79%</strong></td>
</tr>
</tbody>
</table>

Table 4.6.4: Size comparisons of gzip-compressed HMML and BiM compressed HMML against uncompressed HMML, and uncompressed HMML and BiM-compressed HMML against original uncompressed BVH sizes.
mining factor in BVH size. This is a direct consequence of the BVH format, where the position of all parts of the body are constantly recorded within a specified time interval (i.e. 120 frames per second), even when the part in question is not in motion (the BVH format was discussed in detail in Chapter 2).

In contrast, Table 4.6.3 shows the average number of elements in the HMML annotation of each motion type. In terms of the number of elements required to annotate a motion, on average, the motion “jumping jack” contains more HMML elements compared to the other motion types due to the jumping jack motion being more complex compared to the other motions. These more complex motions tend to result in a larger HMML document instantiation, where in this specific example, jumping jack motions generally create HMML documents that are 2.8 times larger than walking forward motions. Hence, unlike BVH, HMML description of a motion is highly dependent on the complexity of the motion. With the ability of HMML to describe only specific limbs of interest in a specific movement direction, queries using HMML do not need to transmit redundant information (i.e. limbs that are not in motion or not of interest are not included in the query). This allow queries using HMML to be smaller compared to using BVH.

Table 4.6.3 also shows the sizes of HMML annotations. In terms of the overall HMML document size compared to the original BVH recordings of a motion, where the average HMML document size is shown to be 99.6% smaller than the size of the original BVH files. This saving in file sizes will be evident if a user performs a query-by-example, where using uncompressed HMML to serve as the example instead of using the original motion recording (BVH files in this scenario) will result in a considerable saving in the size of the query (up to 99.6% smaller query size).

Further, Table 4.6.4 shows various size comparisons of HMML documents compressed using gzip and BiM. Table 4.6.4 also shows the uncompressed HMML document sizes and BiM-compressed HMML as compared to the original uncompressed BVH files. On average, compressing HMML using BiM results in files that are 9.79% the size of the original HMML documents, which is a sizable compression compared to gzip-compressed HMML documents (which on average are 39.13% the size of the uncompressed HMML documents). Compared to the original BVH recordings, uncompressed HMML are typically less than 0.5% the size of the original BVH file. Compressing the HMML descriptions using BiM will generally result in files that are 0.05% the size of the original BVH files.

### 4.7 Conclusions

Multimedia search using existing description formats can be thought of as increasingly strict search filtering based on a Subject-Verb-Object construct, where a non-
temporal Dublin Core can describe the Subject, and temporal MPEG-7 descriptors can describe the Object portions of the construct. However, the important Verb portion is notably missing, which HMML intends to fill. HMML was not designed to replace either Dublin Core or MPEG-7. Instead, it was designed to work in conjunction with existing standards to provide a richer multimedia query environment, where a detailed query such as “John Doe running and jumping into a lake” can be performed with high temporal accuracy. HMML is based on XML Schema, which provide interoperability with any current and future description formats that are based on XML. Key features of HMML include limb-based motion annotation using anatomical body planes that provides a constant frame of reference (independent of the direction that the body is facing), temporal matching features for query construction, and the facilities to match arbitrarily repeated motions (such as leg movements in walking) using a single query term.

HMML was designed primarily for human motion enhanced multimedia search, and also as a standard human motion annotation interchange format that is designed to convey the semantics of a motion. Currently, query-by-example of human motion would require the transmission of motion recordings which are semantic-less and requires a considerable bandwidth to transmit. For this purpose, an ad-hoc motion description using HMML can serve as the example motion when an actual motion recording is not available.

Example descriptions of human movement in three dimensions using HMML were offered in section 4.3 (HMML design description), and query examples were offered in section 4.4 (semantic motion queries using HMML) and 4.5 (query using HMML+MQF). Size comparisons are done in section 4.6 to reflect experiments performed in chapter 3 section 3.6 and to illustrate the overall bandwidth usage of human motion search using HMML.

Results have shown that using HMML in future human motion query-by-example could reduce the query size up to 0.49% compared to using the original recording (such as BVH) as the example motion. However, further processing of the HMML descriptions using data compression techniques will result in much smaller descriptions, especially when using techniques that are specifically designed to compress XML documents such as BiM, where an additional advantage of using BiM is the preservation of the structure of the XML file (even in compressed form).
Chapter 5

Motion Feature Extraction

5.1 Introduction

To facilitate human motion search and description, two required sub-systems were discussed in this thesis; namely, the semantics interchange format in the form of the Human Motion Markup Language (HMML, in Chapter 4), and the communications format in the form of the Multimedia Query Format (MQF, in Chapter 3). Although it was shown in Chapters 3 and 4 that it is possible to perform a query-by-example and query-by-semantics of human motion using both HMML and MQF, the HMML...
annotation that drives the whole process would have to be derived for the source media before any steps toward search can be taken.

As the first step toward the goal of automatic annotation of human motion, feature extraction from motion capture recordings is discussed in this Chapter, where the motion features would have to fulfill the requirements of:

- Comparable features would have to be able to be extracted from comparable motion, where the extracted features would have to reflect the motion being performed. For example, features extracted from one walking motion should be comparable to features extracted from another walking motion.

- The extraction process should allow the filtering of jitters and errors that could occur in a motion recording.

- The features should be able to retain any artifact in the recording if required, thus preserving the uniqueness of a motion recording.

- The resulting features should be translatable to HMML.

This chapter thus presents features that could be automatically extracted from motion capture data towards an automatic extraction of motion description using HMML (shown as the highlighted box in Figure 5.0.1 with respect to the overall system), where the method presented involves automatic derivation from 3D motion capture data using a technique called “partial reconstruction”. This chapter is organized as follows: Section 5.2 will provide details on how the partial reconstruction method is applied to 3D motion capture data and how the semantics of a motion are recognized; Section 5.3 analyzes the consistency of symbols extracted from motion capture data; Section 5.4 describes automatic HMML extraction from motion capture data and will conduct an evaluation of the features extracted from motion capture in motion search; and finally Section 5.5 will provide the concluding remarks of the chapter.

5.2 Extracting Features From Motion Capture Data

Since HMML was designed to describe human motion in three dimensions, the investigation into an automatic HMML extraction process naturally utilizes a 3D representation of human motion (the most accurate representation of which is motion capture). This chapter will investigate feature extraction from an inverse kinematic representation of human motion in the BVH format (as previously described in Chapter 2). Since HMML is designed to represent the movements of the human body in terms of limbs (i.e. the arms, the legs, the backbone, and the head), a
Limbs are extracted from the source media, where the movements in the sagittal, coronal, and transverse planes (in the Plane Separation process) for each limb are detected. The moments between movements are assigned predefined symbols, and the flow from one symbol to the next are recognized as the motion semantics.

In Figure 5.2.1, the source media is separated into recognizable limbs which correspond to the limb definitions in HMML. The next step is to separate the movements of individual limbs into the three orthogonal planes (sagittal, coronal, and transverse). Following this plane separation, the movements of each limb in each plane would have to be detected for changes in the associated direction due to the HMML method of describing only movements, where a movement can be observed to occur when: a limb has stopped moving in one direction and started to move to the opposite direction; and a limb starts to move in a certain direction from a “held” position. For example, a hand that is being held in place can be observed to commence moving upward until the moment upward movement stops. The period between the hand being held in place until it ceases moving upward is the motion...
that corresponds to the HMML description of a motion (see Chapter 4). The mo-
ments of direction change (discussed in Subsection 5.2.2) are assigned predefined
symbols (discussed in Subsection 5.2.3), where the flow of one symbol to the next
signifies the starting and ending moments of a movement, and hence can be assigned
a specific HMML element with known “start” and “end” value (shown in Subsection
5.4.1).

5.2.1 Partial Reconstruction of the Motion Capture Skeleton
Hierarchy

To extract features from human motion that are comparable from one motion to
another, the body would have to be orientated in a consistent fashion. For example,
consider a motion of walking four steps forward, turning the body to face left, and
continued by walking four more steps forward. An observer from behind the per-
former would observe that exact description of motion being performed. However,
an observer from the front of the performer would see a person walking forward
four steps, turning, and then walking to the opposite direction observed by the first
observer (i.e. to the right instead of to the left). Ideally, the motion would be de-
scribed as “walking forward four steps; turning; walking forward four steps”, i.e. the
motion needs to be described from the point of view of the performer and not the
observers. An illustration of this problem is shown in Figure 5.2.2, where in Fig-
ure 5.2.2(a) the reconstructed motion is showing a person walking in a semi-circle
toward the left, and Figure 5.2.2(b) shows the normalized body facing so that the
body always faces a constant direction. If the extracted feature is to be comparable
between motions, processing would have to be done on the original motion data so
that every reconstructed motion would face a constant direction.

Further, the height and the length of the limb of the person performing the
motion would need to be normalized, so that the same motion performed by a tall
person having a longer limb would identify to the same motion as performed by a
short person having a shorter limb.

The field of anthropometry (i.e. the scientific study of the measurements and pro-
portions of the human body) revealed that human measurements are highly variable
in gender, ethnicity, and age, with most measurements following a normal distribu-
tion [125]. However, anthropometry studies are used to design work spaces (e.g.
chairs, tables) and range of reach for designing the most optimal arrangement suit-
able for the majority of the population. Anthropometry does not describe whether a
walking motion is performed similarly or not between people with different measure-
ments; for example, Pheasant [125] described that on average Africans have longer
legs compared to other ethnic groups, and Japanese generally have shorter legs (even
Figure 5.2.2: An illustration of the problem of body facing in feature extraction of motion capture recording. (a) the reconstructed motion shows a person walking in a semi-circle to the left, and (b) normalized motion reconstruction so that the person is facing a constant direction.
among other Asian ethnic groups such as Chinese, Koreans, or Vietnamese). Unfortunately, anthropometric studies do not explain whether Africans and Japanese legs move in the same manner during walking motion, and studies exploring movement similarities between people of different limb length are scarce in the literature.

Recent research on walking motion between different-sized individuals performed by Weyand et al. [126] determined that during walking, the ratio of stride length vs. body height across people with different heights is largely constant (i.e. people with different anthropometry measurements are walking using the same range of motions that are comparable between people with long legs and short legs). The conclusion of [126] is that in general, the movement of the legs during walking is invariant across individuals and the only variable is governed by their body height. In a human motion search system, this variability of limb length from person to person would also need to be removed and the motions described from a consistent representation so that a motion can be described in a similar fashion without regard to the height of the person.

The solution to the two problems described above is twofold: to solve the body orientation problem, the origin point for the reconstruction process would have to be located inside the skeletal hierarchy so that the reconstructed body is aligned to a constant direction; and to solve the limb length problem, a representation that describes limb movements without taking into account any specific length of a limb is required. Fortunately, the limb length problem has been partially solved in the form of dance notations and HMML, where specifically in Labanotation, the movement of a limb is described without information regarding the limb length (see section 2.4.4). Dance notation such as Labanotation cannot exist if anthropometric measurements are essential to the correct description of movements.

The root bone (shown in Section D) contains the information of the rotation and translation of the body as a whole. Therefore, to achieve body orientation normalization, the motion could be reconstructed by setting the root bone to act as the origin point for all subsequent reconstruction process, instead of using an origin point that is not located within the skeletal hierarchy (this change of origin point in reconstruction is possible due to the hierarchical nature of motion capture data), so that the global $x$ axis will always describe movements in the coronal plane, the global $y$ axis will always describe movements in the transverse plane, and the global $z$ axis will always describe movements in the sagittal plane. Reconstructed in this manner, the point of view of the reconstructed motion is consistently aligned with the point of view of the performer instead of an observer (i.e. similar to how a motion is described in a notational method such as Labanotation [78]), hence making the movements of the limbs comparable between one motion to another motion, even if the motion involves the turning of the body.
To ensure that individual limb movements are comparable from one motion to another, a limb (left arm, right arm, left leg, right leg, torso, head) is defined in this work as the collection of bones that form the limb from its extremity to a point where the limb connects to the body (for example, the arm limb-unit consists of the bones of the upper arm, lower arm, the hand, and the fingers). In motion capture formats utilizing the typical BVH skeleton hierarchy as shown in Figure 5.2.3, bones that participate in partial reconstruction are highlighted in yellow, while the bone that does not is uncolored. For example, since a limb can originate from one extremity (e.g., the shoulder joint of an arm) to another (e.g., the fingers of that same arm, where the thumb is not the extremity of the arm and therefore not involved in the limb reconstruction process), the arm therefore consists of bones of the: clavicle, humerus, radius, wrist, and fingers.

The result of individual reconstruction of limbs as illustrated in Figure 5.2.3 is that each limb is described in its own local space (see Section 2.4.6 in Chapter 2) that is a combination of all the local spaces from each bone that makes up a limb. Therefore, each limb has one end at the origin of the combined local space, and
CHAPTER 5. MOTION FEATURE EXTRACTION

<table>
<thead>
<tr>
<th>Limb</th>
<th>Bones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left arm</td>
<td>Left clavicle - Left humerus - Left radius - Left fingers</td>
</tr>
<tr>
<td>Right arm</td>
<td>Right clavicle - Right humerus - Right radius - Right fingers</td>
</tr>
<tr>
<td>Left leg</td>
<td>Left hip - Left femur - Left tibia - Left foot - Left toe</td>
</tr>
<tr>
<td>Right leg</td>
<td>Right hip - Right femur - Right tibia - Right foot - Right toe</td>
</tr>
<tr>
<td>Torso</td>
<td>Lowerback - Upperback - Thorax - Lowerneck - Uperneck</td>
</tr>
<tr>
<td>Head</td>
<td>Head</td>
</tr>
</tbody>
</table>

Table 5.2.1: Limb makeup of motion capture partial reconstruction.

Figure 5.2.4: Partially reconstructed left leg and its extremity.

the end-effector of the limb is the furthest point of the combined local spaces. For example, the extremity of the left leg limb is the position of the left toe with respect to the left hip (illustrated in Figure 5.2.4).

After partial reconstruction, it is then possible to extract the movements of the limb as a unit irrespective of the movements of other limbs or the body as a whole, and to compare its movements to the same limb in another motion, since now the limbs are facing a constant direction (i.e. from the point of view of the performer). To extract features that are comparable between motions, the feature extraction technique consists of three steps:

1. Determining the angular values that are created by the limb with respect to the root bone, which is the origin point of a reconstructed limb (in Subsection 5.2.2).
2. Assigning symbols to the reconstructed angular values (in Subsection 5.2.3).

3. Removal of movements under a certain angular value to facilitate jitter removal from the motion data (in Subsection 5.2.4).

5.2.2 Determining the Angular Values

Results presented by Weyand et al. [126] concluded that the walking motion is performed with high consistency from a biological (i.e. movement) point of view between different individuals, since the goal of a walking motion is to move the body from one point to another using as little energy as possible. Any apparent differences in walking motions between individuals are governed by the person’s stature (i.e. height and the length of their limbs). Weyand observed that to achieve the most efficient energy usage in walking, the stride length of a person is directly correlated to the length of their limb. Therefore, the angles created by the femur-toe unit (i.e. the legs) relative to the body are relatively constant across different individuals in a walking motion. Therefore, eliminating the variability of body height is possible by determining the angles that a limb unit creates with respect to a specific body part that is assumed to be static (such as the angles the femur-tibia-foot-toes unit with respect to the hips). Hence, the motions of a limb can be recognized by observing the angles created during that motion.

For the legs, the femur-tibia-foot-toes unit is locally reconstructed and the bones of the legs (femur-tibia-foot-toes; illustrated in Figures 5.2.3 and 5.2.4) are considered as being joined into one unit (which was previously defined in Section 5.2.1 as a limb). The coordinates of the toes relative to the femur projected into either the sagittal or coronal plane would provide the location of the toes relative to the femur. Projecting the reconstructed limb into the sagittal plane would provide the forward-backward movements and the coronal plane would provide side-to-side movements. The projection into the three anatomical planes is performed using:

- sagittal plane:
  \[
  \alpha_{\text{Sagittal}} = \arctan^2(z, y) \tag{5.2.1}
  \]

- coronal plane:
  \[
  \beta_{\text{Coronal}} = \arctan^2(x, y) \tag{5.2.2}
  \]

- transverse plane: the transverse plane is a special case where the movement in this plane is represented by the lengthening/shortening of the vector \( \overrightarrow{OP} \):
Figure 5.2.5: Projection of a limb represented by the vector $\overrightarrow{OP}$ into its sagittal (the angle $\alpha$) and coronal (the angle $\beta$) plane components. The transverse plane component is represented by the length of the vector $\overrightarrow{OP}$. 
\[ r_{\text{Transverse}} = \sqrt{x^2 + y^2 + z^2} \]  

(5.2.3)

Where the function \( \text{atan}2() \) in 5.2.1 and 5.2.2 is the four-quadrant version of the inverse tangent function \([127]\). The function \( \text{atan}2() \) is used due to the definition of the inverse tangent \( (\text{atan}()) \) function, where the \( \text{atan}() \) function is only defined for two quadrants (most commonly \((0, \pi)\)) instead of all four quadrants as required by the partial reconstruction technique. The function \( \text{atan}2() \) is therefore used since some limbs e.g. the arms have a movement range of 360 degrees in the sagittal plane.

For example, if the location of the toes relative to the hips in the cartesian coordinates is \((0,5,3)\) (i.e. 0 units in the coronal plane, 5 units in the transverse plane, and 3 units in the sagittal plane, i.e. the leg is moving forward relative to the hips) where a unit is defined in individual BVH file, then the angles created by the leg are as illustrated in Figure 5.2.6:

- **sagittal plane**: \( \alpha = \text{atan}2(z, y) = \text{atan}2(3, 5) = 30.9 \) degrees
- **coronal plane**: \( \beta = \text{atan}2(x, y) = \text{atan}2(0, 5) = 0 \) degrees
- **transverse plane (length of vector \( \overrightarrow{OP} \))**: \( r = \sqrt{x^2 + y^2 + z^2} = \sqrt{0^2 + 5^2 + 3^2} = 5.8 \) units

Therefore the leg unit (viewed from the sagittal plane) creates an angle of 30.9 degrees forward, with the whole leg unit having a length of 5.8 units. This process is repeated for every frame of the motion. Plotting the angular values of \( \alpha \) with respect to time for the left and right legs results in Figure 5.2.7. Note that Figures 5.2.5 and 5.2.6 uses the same \( \text{atan}2 \) function, however the Figures are showing the function operating in different quadrants.

By examining the angular values plot in Figure 5.2.7, certain features of the reconstructed motion can be observed: the angle created by the legs relative to the body during leg-forward motion (at most 30 degrees of variation of the sagittal plane/y-axis projection), and the steps taken during walking (five steps in total starting with the right leg, from the number of peaks in Figure 5.2.7).

### 5.2.3 Symbolization of the Angular Values

The next step in the motion feature extraction technique is to partition the angular plot into recognizable segments. For example, a walking motion segment for the left leg might be defined as that which contains the moment from the left leg swinging forward up to the moment where it makes non-sliding contact with the ground to receive the body weight (one step). Conversely, a sub-motion segment for the right leg would consist of it swinging forwards up to the moment where the right leg
CHAPTER 5. MOTION FEATURE EXTRACTION

Figure 5.2.6: 3D view of a leg unit projected into the sagittal and coronal plane. Point O represents the origin of the limb unit (in this case the hip) and point E represents the extremity of the limb unit (in this case the foot). The limb unit is projected into the two anatomical planes (sagittal and coronal).
receives the body weight (the second step). The two types of segments of a walking motion are shown in Figure 5.2.8 with respect to the angular plot. In Figure 5.2.8, the two sub-motion segments (left leg and right leg forward step) occurred two times since the motion is a “walking forward five steps”. The first step was not recognized in the plot in Figure 5.2.8 since it’s not a fully complete step (i.e. the motion recording did not record the moment where the right leg swings forward adequately).

In order to successfully segment the motion, the positive and negative peaks (i.e. the turning points) from the angular plot need to be determined. The maximum peaks or minimum troughs in Figure 5.2.7 indicates when each limb has stopped moving in a particular direction. Therefore, using the walking motion as an example, a motion of “right leg stepping forward” is defined as a sequence of: one trough of the right leg; one peak of the right leg; and one trough of the opposite leg (the left leg in this example).

Figure 5.2.7 contains two data lines: the left and the right leg angular values in the sagittal plane. The peak and troughs in Figure 5.2.7 can be uniquely identified and assigned a unique symbol (one symbol for a peak and one symbol for a trough), which in Figure 5.2.7 results in four symbols that can be associated with the left and right leg movements in the sagittal plane. The labeled plot of Figure 5.2.7 is shown in Figure 5.2.9, and the symbols are:

- Left leg maximum/peak ($a$)
- Left leg minimum/trough ($b$)
Figure 5.2.8: Correlation between a plot of reconstructed angular values in a walking motion to the sub-motion of walking. In this example, the two sub-motions are one forward step with the left leg and one forward step with the right leg, where a “forward step” is defined from the moment where the leg swinging forward up to the moment where it makes non-sliding contact with the ground to receive the body weight.

Figure 5.2.9: Symbolization of the peak and throughs of the angular plot for the legs for a walking motion in the sagittal plane. The left leg peak is symbolized as $a$; the left leg through is symbolized as $b$; the right leg peak is symbolized as $c$; and the right leg through is symbolized as $d$. 
<table>
<thead>
<tr>
<th>Limb</th>
<th>Side</th>
<th>Plane</th>
<th>Movement</th>
<th>Symbol From</th>
<th>Symbol To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leg</td>
<td>Left</td>
<td>sagittal</td>
<td>Forward</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Backward</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>Leg</td>
<td>Right</td>
<td>sagittal</td>
<td>Forward</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Backward</td>
<td>d</td>
<td>c</td>
</tr>
<tr>
<td>Leg</td>
<td>Left</td>
<td>coronal</td>
<td>Right</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Left</td>
<td>f</td>
<td>e</td>
</tr>
<tr>
<td>Leg</td>
<td>Right</td>
<td>coronal</td>
<td>Right</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Left</td>
<td>h</td>
<td>g</td>
</tr>
<tr>
<td>Leg</td>
<td>Left</td>
<td>transverse</td>
<td>Down</td>
<td>i</td>
<td>j</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Up</td>
<td>j</td>
<td>i</td>
</tr>
<tr>
<td>Leg</td>
<td>Right</td>
<td>transverse</td>
<td>Down</td>
<td>k</td>
<td>l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Up</td>
<td>l</td>
<td>k</td>
</tr>
<tr>
<td>Arm</td>
<td>Left</td>
<td>sagittal</td>
<td>Forward</td>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Backward</td>
<td>n</td>
<td>m</td>
</tr>
<tr>
<td>Arm</td>
<td>Right</td>
<td>sagittal</td>
<td>Forward</td>
<td>o</td>
<td>p</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Backward</td>
<td>p</td>
<td>o</td>
</tr>
<tr>
<td>Arm</td>
<td>Left</td>
<td>coronal</td>
<td>Right</td>
<td>q</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Left</td>
<td>r</td>
<td>q</td>
</tr>
<tr>
<td>Arm</td>
<td>Right</td>
<td>coronal</td>
<td>Right</td>
<td>s</td>
<td>t</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Left</td>
<td>t</td>
<td>s</td>
</tr>
<tr>
<td>Arm</td>
<td>Left</td>
<td>transverse</td>
<td>Down</td>
<td>u</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Up</td>
<td>v</td>
<td>u</td>
</tr>
<tr>
<td>Arm</td>
<td>Right</td>
<td>transverse</td>
<td>Down</td>
<td>w</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Up</td>
<td>x</td>
<td>w</td>
</tr>
</tbody>
</table>

Table 5.2.2: Correlation between feature symbols and limb movements (arms and legs).
Table 5.2.3: Correlation between feature symbols and limb movements (backbone and head). Note that the symbols associated with the turning points are case-sensitive.

- Right leg maximum/peak (c)
- Right leg minimum/trough (d)

Symbolizing all possible combinations of the peak and troughs for all limbs (i.e. left arm, left leg, right arm, right leg, backbone and head) and all anatomical planes (sagittal, coronal, and transverse) results in Tables 5.2.2 and 5.2.3.

By arranging the symbolized peak and troughs in Figure 5.2.7 by the order of occurrence in the time domain and examining the resulting sequence of the symbols, a correlation between the two sub-motions of walking (in Figure 5.2.8) and the assigned symbols (in Figure 5.2.9) can be observed:

- Left leg forward step: a-b-c
- Right leg forward step: c-d-a

By using the symbol list in Tables 5.2.2 and 5.2.3, a symbol stream can be extracted from a walking motion that includes the arms as well as the legs, where the workflow from the motion recording to the resulting symbols is shown in Figure 5.2.10 (only the sagittal plane symbols are shown for brevity).

In Figure 5.2.10, the extracted symbol stream for the arms is n-o-p-m-n-o-p-m, and for the legs is c-d-a-b-c-d-a-b-c. Combining the two sequences together and
Figure 5.2.10: Detailed workflow of the symbol stream extraction process in the sagittal plane of a walking motion. The resulting symbol stream in this example is c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c. Specifically, the symbol stream generated is c-d-a-b-c-d-a-b-c and n-o-p-m-n-o-p-m for the legs and the arms, respectively.
arranging the symbols in the time domain results in the symbol sequence of $c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c$. Taking the sequence $a-b-c$ (left leg stepping forward), it can be observed that in the symbol sequence of $c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c$ there exist two instances where the sequence $a-b-c$ occurs (with other assorted symbols occurring in-between), which corresponds to the two instances where the left leg steps forward.

5.2.4 Movement Detection Threshold (MDT)

One important aspect of the feature extraction process described in Section 5.2.2 is the question of the magnitude of direction change that must occur before it is recognized as a symbol. For example, if the left arm was detected to be moving a single degree, it is hard to determine if this one degree of movement is an actual movement or a jitter that occurred during the recording process. One possible solution to eliminate this jitter is to provide a variable that governs how much movement is to occur before a turning point is symbolized. This variable is called “Movement Detection Threshold (MDT)”, where the MDT value is used in a function $R$ that is defined as:

$$
R(MDT, a_i, a_{i-1}) = \begin{cases} 
1 & \text{if } a_i - a_{i-1} \geq MDT, MDT \geq 0 \\
0 & \text{otherwise}
\end{cases}
$$

(5.2.4)

where $MDT$ is the MDT value, $a_i$ is the current turning point being evaluated, and $a_{i-1}$ is the value of the previous turning point. The result of $T$ would be a binary mask that would determine whether a turning point is to be recognized ($R=1$) or discarded ($R=0$). Therefore, function $R$ specifies how much detail is to be “symbolized” in a motion.

An illustration of the process of MDT in Equation (5.2.4) is shown in Figure 5.2.11. In Figure 5.2.11, if the value is set at 0, then all turning points labelled $x$, $y$, and $z$ are recognized as features and recorded as their corresponding symbols. However, if the MDT value is set at 2, then only points $x$ and $z$ would be recognized, since the difference between points $x$ and $y$ is smaller than 2 degrees, and the difference between points $y$ and $z$ is larger than 2 degrees. An example of the effect of MDT in a symbol sequence fragment using values from 0 to 20 is shown in Figure 5.2.12, where the motion being performed is ‘walking in an anti-clockwise circle’.

Since the MDT value determines the magnitude of turning points that are filtered in or out, this also reveals information about the motion – symbols that consistently appear at various MDT values indicate that they are larger and more significant to that motion. For example, consider the example shown in Figure 5.2.12. Figure 5.2.12 shows that all of the symbols that were extracted using an MDT value of 20,
Figure 5.2.11: An illustration of the use of the Movement Detection Threshold (MDT) value. If the MDT value is set at 0, then points \( x, y, \) and \( z \) are all recognized as valid turning points. However if the MDT value is set to 2, point \( y \) would not be recognized as a valid turning point since the difference between \( x \) and \( y \) is less than the MDT value, but point \( z \) will be recognized since the difference between point \( z \) and a previous turning point (even if it’s not included in the list of valid turning points, which is point \( y \) in this case) is more than the MDT value.

exist within the set of symbols that were extracted using all MDT values lower than 20. Figure 5.2.12 also shows that that the motions in the sagittal (i.e., forward and backward) plane are the largest features; however, since the motion in Figure 5.2.12 is ‘walking in an anti-clockwise circle’, symbols representing side-to-side movements of the right leg are also present – these features are \( c-h-n-d-g-a-b \) which correspond to: Right Leg sagittal \( (c) \to \) Right Leg coronal \( (h) \to \) Left Arm sagittal \( (n) \to \) Right Leg sagittal \( (d) \to \) Right Leg coronal \( (g) \to \) Left Leg sagittal \( (a) \to \) Left Leg sagittal \( (b) \) occur in that order. Hence, the set of symbols at each MDT value is a subset of the set of symbols at lower MDT values, and can be expressed as:

\[
f(MDT + 1, m) \subset f(MDT, m), t \in \{0, N\}
\]

(5.2.5)

where \( f(t, m) \) is the symbolization process of motion \( m \) using a certain MDT value. Thus the significance of the MDT value is that it provides the size or “weighting” of the features.

Since Equation (5.2.5) shows that a higher MDT values are subsets of lower MDT values (e.g. MDT value of 1 is always a subset of MDT values >1), one question is: How much smaller is the number of symbols extracted at higher MDT values with respect to lower values? To examine this question, symbols were extracted from the HDM database [123] which contains 270 motion recordings having an average length of 41 seconds. Table 5.2.4 shows the total number of symbols that are extracted from the HDM database using MDT values of 0-20, and also shows the ratio between the
CHAPTER 5. MOTION FEATURE EXTRACTION

![Image of motion sequence and extracted symbols]

Figure 5.2.12: The effect of different MDT values on a fragment of extracted symbol sequences for a walking motion.

<table>
<thead>
<tr>
<th>MDT value</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>chfekxldkuiwfoxljgweraqxvtevb</td>
</tr>
<tr>
<td>1</td>
<td>chfekxldkuiwfoxljgerafitvqekb</td>
</tr>
<tr>
<td>2</td>
<td>chfekldkuiwfoxljgerafitvqekb</td>
</tr>
<tr>
<td>3</td>
<td>chfekldkuiwfoxljgerafitvqekb</td>
</tr>
<tr>
<td>4</td>
<td>chfe nd fo g af iteb</td>
</tr>
<tr>
<td>5</td>
<td>chfe nd fo g af iteb</td>
</tr>
<tr>
<td>6</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>7</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>8</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>9</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>10</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>11</td>
<td>chf nd o g af ieb</td>
</tr>
<tr>
<td>12</td>
<td>ch nd o g af ieb</td>
</tr>
<tr>
<td>13</td>
<td>ch nd o g af ieb</td>
</tr>
<tr>
<td>14</td>
<td>ch nd o g af ieb</td>
</tr>
<tr>
<td>15</td>
<td>ch nd o g af ieb</td>
</tr>
<tr>
<td>16</td>
<td>ch nd o g af ieb</td>
</tr>
<tr>
<td>17</td>
<td>ch nd o g a b</td>
</tr>
<tr>
<td>18</td>
<td>ch nd o g a b</td>
</tr>
<tr>
<td>19</td>
<td>ch nd o g a b</td>
</tr>
<tr>
<td>20</td>
<td>ch nd o g a b</td>
</tr>
</tbody>
</table>
Table 5.2.4: The total number of symbols extracted from the whole HDM motion capture database using different MDT values.

<table>
<thead>
<tr>
<th>MDT value</th>
<th>Total number of symbols</th>
<th>Ratio to MDT=0</th>
<th>MDT value</th>
<th>Total number of symbols</th>
<th>Ratio to MDT=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>540,788</td>
<td>1.00</td>
<td>11</td>
<td>128,629</td>
<td>0.24</td>
</tr>
<tr>
<td>1</td>
<td>334,758</td>
<td>0.62</td>
<td>12</td>
<td>119,867</td>
<td>0.22</td>
</tr>
<tr>
<td>2</td>
<td>284,781</td>
<td>0.53</td>
<td>13</td>
<td>112,381</td>
<td>0.21</td>
</tr>
<tr>
<td>3</td>
<td>254,648</td>
<td>0.47</td>
<td>14</td>
<td>105,845</td>
<td>0.20</td>
</tr>
<tr>
<td>4</td>
<td>230,818</td>
<td>0.43</td>
<td>15</td>
<td>99,967</td>
<td>0.18</td>
</tr>
<tr>
<td>5</td>
<td>210,235</td>
<td>0.39</td>
<td>16</td>
<td>94,783</td>
<td>0.18</td>
</tr>
<tr>
<td>6</td>
<td>191,982</td>
<td>0.36</td>
<td>17</td>
<td>90,128</td>
<td>0.17</td>
</tr>
<tr>
<td>7</td>
<td>175,946</td>
<td>0.33</td>
<td>18</td>
<td>85,764</td>
<td>0.16</td>
</tr>
<tr>
<td>8</td>
<td>161,973</td>
<td>0.30</td>
<td>19</td>
<td>81,829</td>
<td>0.15</td>
</tr>
<tr>
<td>9</td>
<td>149,665</td>
<td>0.28</td>
<td>20</td>
<td>78,095</td>
<td>0.14</td>
</tr>
<tr>
<td>10</td>
<td>138,592</td>
<td>0.26</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Although extracting the symbol sequences at MDT values >0 implies that some turning points are not recorded as a symbol (thus having a less detailed symbol sequence), an advantage of extraction using a higher MDT value is the amount of storage space necessary to store and/or transmit the symbol sequences. For example, referring to Table 5.2.4, if symbol sequences for the whole HDM motion capture database were extracted and stored, extracting at MDT value of 0 would require at least 540,788 bytes if one symbol is stored as one byte. In contrast, increasing the MDT value to 1 resulted in 62% of the number of symbols compared to MDT value of 0 (a decrease of 38% of the number of symbols that needs to be stored/transmitted). At an MDT value of 20, only 14% of the symbols are retained (which corresponds to 86% decrease in the number of symbols vs. MDT value of 0); however, an MDT value of 20 also means that only movements of ≥20 degrees are symbolized.

In light of the result showing the total number of symbols extracted using various MDT values in Table 5.2.4, another question is how much does the number of extracted symbols vary with respect to the MDT value? To answer this question, the same HDM database was again used for testing, where the mean and 95% confidence interval of the number of symbols extracted were calculated per file in the HDM database. The result is shown in Figure 5.2.13.

Figure 5.2.13 plots the mean of the number of extracted symbols per file, with error bars representing the 95% confidence interval. Similar to the result in Table
Figure 5.2.13: The average number of features extracted per file for MDT values of 0-20 for the HDM motion capture database. Total number of files is 270, where the average length of each file is 41 seconds. Error bars in the graph represent the 95% confidence interval.
5.2.4, the number of extracted symbols drop significantly from an MDT value of 0 to an MDT value of 1. In Figure 5.2.13, the average number of symbols at an MDT value of 0 is 2022.92 (±113.42), and for an MDT value of 1 is 1239.84 (±68.44). The large discrepancy between MDT values of 0 vs. 1 is further reinforced by the large gap between the error bars, which shows that on average, extracting at an MDT value of 1 will generally results in 62% of the number of symbols extracted at an MDT value of 0. Subsequent MDT values of 2-20 show a trend of reducing error bars, to the point where the error bars starting to overlap at MDT values beyond 3. However, the ratio of the range of the error bar with respect to its mean value across all MDT values tested never goes beyond 0.06 (the highest ratio of mean to confidence interval is 0.0595 for MDT value of 20), which shows that the number of extracted symbols/file is relatively consistent across all files for a specific MDT value.

5.2.5 Symbol Sequence Uniqueness

Motion capture is a highly detailed recording of a human motion, and symbol sequence extraction attempts to provide a feature set that accurately represents the nuances of the motion in the recorded data. Since motion capture is a recording in time, no two motion capture files are exactly identical (since the same movements being performed by a person will have slight differences during each performance). However, results shown in Table 5.2.4 and Figure 5.2.13. show that the MDT value used during symbol extraction provides a way to minimize the number of extracted symbols, thus saving storage space in the process. Therefore, an investigation into the lowest MDT value that still retains the uniqueness of the motion capture recordings (i.e. comparable level of detail in motion capture data vs. the symbol sequences) is a question that will be addressed in this section.

Experiment Methodology

The MDT value to be used in this uniqueness experiment are based on the results of the analysis of the statistics of various MDT values shown in Figure 5.2.13 in Section 5.2.4 (i.e. MDT values of 0, 1, 5, 10, 15, and 20). For scoring the experiment, precision and recall scores are used:

\[
\text{Precision} = \frac{tp}{tp + fp} \quad (5.2.6)
\]

\[
\text{Recall} = \frac{tp}{tp + fn} \quad (5.2.7)
\]

where in (5.2.6) and (5.2.7), \(tp\) is the true positives (correct result marked as
correct), \( fp \) is the false positive (false result marked as correct), and \( fn \) is the false negative (correct result marked as false).

For this experiment, the motion database utilized was the HDMcuts database [123], which consists of 852 motion files, totaling 43.7 minutes of motion data in 60 types of motions captured at 120 frames per second. The files in this database are grouped into single types of motion (e.g., walking, jogging, kick, punch, etc.) performed by one subject per file, where each type was performed multiple times by five different subjects. The files in the database are manually prepared by the database authors, hence even among the same motion type, each file is unique. The average length of the motion files is 3.1 seconds.

Additionally, the experiment was performed using different “window” sizes, where the input symbol sequence from the start of the motion was cut after 1 second of motion, 2 seconds of motion, 3 seconds of motion, and finally 4 seconds of motion. The maximum window size of 4 seconds was used since the average length of the motion files is 3.1 seconds. The goal of using a windowing scheme is to determine the minimum length of motion necessary that is still uniquely identifiable by using the symbol sequence method.

Experiment Results

The results for using the same threshold for the codebook and the query are shown in Figure 5.2.14. The x-axis in Figure 5.2.14 shows the MDT value used during the symbol sequence extraction process, and the y-axis shows the precision and recall scores for each MDT value. For all results plotted in Figure 5.2.14, the error bars show the 95% confidence interval of the precision and recall scores.

In Figure 5.2.14, using only 1 second of motion with MDT values of 0, 1, and 5 results in scores of 100% for precision and 100% for recall which is the maximum value for precision and recall possible. Using higher MDT values (i.e. 10, 15 and 20) with one second of motion result in scores of 0.35 (precision) and 0.37 (recall) for MDT value of 10, 0.17 (precision) and 0.19 (recall) for MDT value of 15, and finally 0.12 (precision) and 0.14 (recall) for MDT value of 20. Therefore, an MDT value of 5 to extract symbol sequence from one second of motion still retains the uniqueness of each motion recording.

As the window size is increased to two seconds of motion, maximum precision and recall scores are retained for MDT values of 0, 1, and 5. Improved scores are obtained with MDT value of 10, with scores of 0.98 (precision) and 0.99 (recall). The larger MDT values are still producing low precision and recall scores. Therefore, uniqueness of the symbol sequences are still retained by extracting the symbol sequences of MDT values up to 10 when 2 seconds of motion is being considered.

At window sizes of 3 and 4 seconds, extraction at all MDT values tested (up
Figure 5.2.14: Plots showing the precision and recall values of the matching algorithm using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x axis shows the MDT value of both the input symbol sequence and the codebook symbol sequence was extracted at. Error bars represents 95% confidence intervals.
to 20) are showing improvements in their precision and recall scores beyond 0.95 up to the point of being comparable against the lower MDT values of 0, 1, and 5. Therefore, uniqueness of the symbols are retained throughout all of the MDT values tested when 3 or more seconds of motions are being considered.

In conclusion, an MDT value of 5 or below retains the uniqueness of the recorded motion even for a short motion duration of 1 second. An MDT value of 5 produced only 39% of the number of symbols compared to an MDT value of 0 while at the same time still provides comparable uniqueness. For applications requiring the uniqueness of the recorded motion to be reflected in the extracted symbol sequences while maintaining minimal sizes of the symbol sequences, using an MDT value of 5 provides the best performance.

5.3 Symbol Sequence Timing Analysis

After reviewing the MDT value where each recorded motion can be differentiated uniquely, this section analyzes the similarities between the symbol sequences that were extracted from visually similar motion. For example, two recorded walking motion are determined to be uniquely identifiable (e.g. do not produce the same symbol sequence) when MDT values of 5 or less are used during the extraction process. However, what are the similarities between the two walking motions that is reflected in the extracted symbol sequences? To answer this question, in this section, symbols are extracted from different motions to discover any similarities between the symbol sequences and the motions being performed. Subsection 5.3.1 compares symbols extracted from four motion classes: walking, sneaking, running, and walking backward. Subsection 5.3.2 analyzes the consistency of the extracted symbols from walking motions performed by four different person with different heights and different walking patterns, and Subsection 5.3.3 analyzes the possibility of differentiating between motions having identical filtered symbol sequences.

5.3.1 Filtering The Symbol Sequences

The first part of the answer to the question of the similarities in the symbol sequences between visually similar motions is to be answered by extracting the symbol sequence of eight motions: two walking forward, two walking backward, two sneaking, and two running. In these eight motions, walking forward, running, and sneaking contains visually similar leg movements, i.e. the legs step forward in turn to provide forward locomotion to the body. The walking backward motion contains a different motion visually, therefore any similarities in the preceding three motion classes should not be present in the walking backward motion.
Table 5.3.1: Examples of symbol sequences extracted from walking forward, walking backward, sneaking, and running motions. The extracted symbols contain symbols from all limbs and all anatomical planes. Filtering the extracted symbols to remove all symbols except the ones that represent the leg movements in the sagittal plane reveals the similarities between walking, sneaking, and running motions: the sequence b-c-d-a. The two walking backward motions contain the sequence c-b-a-d instead.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Extracted Symbols (MDT value 0)</th>
<th>Leg Only Symbols (sagittal Plane)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk forward #1</td>
<td>ljbibshkAmFjlrBcxvtdfgkgKiF</td>
<td>bcda</td>
</tr>
<tr>
<td></td>
<td>AqCEsnHlwDduLjFIBKlrxEaeJLG</td>
<td></td>
</tr>
<tr>
<td>Walk forward #2</td>
<td>pnrkbHuBfjsevkxhDfGkgAEquiCn</td>
<td>bcda</td>
</tr>
<tr>
<td></td>
<td>wlkDFHdhtolsJlxlBkLaegrEvJi</td>
<td></td>
</tr>
<tr>
<td>Walk backward #1</td>
<td>gGpwekxoBeFhlhDfnjsQkIkwubHpLg</td>
<td>cbad</td>
</tr>
<tr>
<td></td>
<td>JeEAlKthiGvmfHrqFoEaCGexsgBJFjtL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rihuEdpHnsfDA</td>
<td></td>
</tr>
<tr>
<td>Walk backward #2</td>
<td>HACwukDLjtrheiKcCGBFfjJ</td>
<td>cbad</td>
</tr>
<tr>
<td></td>
<td>fjDLxosvuktKibCgeJLAEmHDfrvhiswG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gFqaBKIejxCiunEdpHlsdHlF</td>
<td></td>
</tr>
<tr>
<td>Sneak #1</td>
<td>vElupGwjemgkbsvqBlKFIcHDAi</td>
<td>bcrf</td>
</tr>
<tr>
<td></td>
<td>LkKtusnGfrLkiHjdaxoEitjckCBLaHIJ</td>
<td></td>
</tr>
<tr>
<td>Sneak #2</td>
<td>lfvqoksjgeGKwbBuLmrxFflItpecw</td>
<td>bcd</td>
</tr>
<tr>
<td></td>
<td>JDLvCHskqAilKDuEnvoxlGhBurjakTFC</td>
<td></td>
</tr>
<tr>
<td>Run #1</td>
<td>rqwlCFmBbjGusHvIxEikAhr</td>
<td>bcd</td>
</tr>
<tr>
<td></td>
<td>DmujuwfotKfbGhJxAE</td>
<td></td>
</tr>
<tr>
<td>Run #2</td>
<td>CliqGmBFKbjtucLvsEIKxiAhk</td>
<td>bcd</td>
</tr>
<tr>
<td></td>
<td>DrHlmuKJgoFtBfdwlaxEJkAvi</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.3.1 shows the result of the extraction process. The extraction process follows the workflow depicted in Figure 5.2.10 using an MDT value of 0 to extract all detected symbols (which could include errors and jitters in the recordings). The extracted symbol sequences shown in Table 5.3.1 contains symbols from all limbs and all anatomical planes arranged in the order of occurrence of the detected turning points. However, since the symbols are unique for each limb and each anatomical plane, it is possible to filter the extracted symbols for the leg movements in the sagittal plane (the symbols for which are \( a, b, c, \) and \( d \)). Filtering out all other symbols resulted in the sequences of \( b-c-d-a \) for walking forward, sneaking, and running motions, and the sequences of \( c-b-a-d \) for the walking backward motions.

Therefore from Table 5.3.1 it can be determined that visually similar motion (the legs stepping forward to provide forward locomotion to the body) does indeed produce the same symbol sequences when a specific limb and anatomical plane that defines the motion is filtered. The next question to be answered is: although walking, running, and sneaking all contain forward locomotion of the body, for most people they are not considered the same motion. How can the symbol sequences differentiate between those three motions? To answer this question, first the consistency of the filtered sequences would have to be determined (i.e. does walking motion always produce the same leg sequence amongst different people and walking pattern?) and secondly, does the time when the symbol occurs provide useful additional information to help determine the difference between walking, running, and sneaking?

### 5.3.2 Symbol Sequence Consistency For Walking Motion

This section investigates the consistency of the extracted symbol sequences from people of different height and limb length. For this investigation, a motion database consisting of walking and running of four subjects are recorded using the Moven motion capture suit [91] inside a room with a 5m x 5m area (designated for the participants to perform motions during the motion capture session) in the University of Wollongong in 2011. The four subjects are:

1. Subject K, male, height: 167 cm.
2. Subject M, male, height: 190 cm.
3. Subject C, male, height: 185 cm.
4. Subject R, female, height: 147 cm.
The subjects are only instructed to perform a walking motion for 30 seconds, without any instruction on how the motion should be performed within the 5m x 5m area. This lack of instruction results in three different walking patterns among the subjects:

1. Subject K walked in a counter-clockwise pattern.
2. Subject M walked in a clockwise pattern.
3. Subject C walked in a random pattern.
4. Subject R walked in a counter-clockwise pattern.

Further, no post-processing (i.e. cleanup or jitter removal) was performed in the motion capture recording.

Figure 5.3.1 shows the result of the experiment, where the resulting symbol sequences shows the $b$-$c$-$d$-$a$ pattern which signifies legs stepping forward in the sagittal plane. This pattern shows up consistently even when the height of the person and the walking patterns are different.

Further, Figure 5.3.2 shows a plot of the amount of elapsed time between the occurrence of symbols in the $b$-$c$-$d$-$a$ pattern:

- The elapsed time from the occurrence of the symbol $b$ to the the occurrence of the symbol $c$;
- From the same $b$ to the occurrence of $d$; and
- From the same $b$ to the occurrence of $a$.

The data in Figure 5.3.2 shows that across the four subjects, the average elapsed time between symbols are quite consistent, where on average:

- Elapsed time from $b$ to $c$ is 0.2 seconds;
- Elapsed time from $b$ to $d$ is 0.65 seconds; and
- Elapsed time from $b$ to $a$ is 0.85 seconds.

From Figures 5.3.1 and 5.3.2, it was shown that the symbol sequences and the timings in an uncleaned (i.e. no jitter or error-removal) motion capture data of four subjects of different height and walking style, the extracted symbol sequence are consistent in the symbol itself and the time of occurrence of the symbols. This result confirms the observations performed by Weyand et al. [126] (discussed in 5.2.1), that is, due to the energy minimization goal of a walking motion, the motion is performed in a consistent manner across individuals, where the differentiating
Figure 5.3.1: Result of an experiment to examine the consistency of symbol sequences extracted from walking motion of four subjects of different heights and different walking patterns. The extracted symbol sequences for the four subjects show a consistent pattern in the symbol sequence of the leg movements in the sagittal plane (b-c-d-a, highlighted in red).
Figure 5.3.2: Timing analysis of each symbol occurrence in the \( b-c-d-a \) pattern in a walking motion shown in Figure 5.3.1. The timings of the symbol occurrences in walking motion show a relatively consistent between subjects of different heights.

factor is the limb length. The relatively consistent timing of symbol occurrence in the result also shows that the symbolized angular values is a consistent and comparable feature set for human motion, even if the symbols are extracted from persons of different heights, walking style, walking pattern, and gender.

5.3.3 Motions With Identical Filtered Sequences

Although Section 5.2.4 shows that the extracted symbols at MDT value of 0 are unique to one specific recording of a motion, post-processing of the symbol sequences to retain symbols from limbs of interest (i.e. only the leg movements in the sagittal plane) could result in identical sequences being produced by different motions having a similar sub-motion such as walking vs. running. This is evident in Table 5.3.1, where Table 5.3.1 shows that for three classes of motions (walking, running, and sneaking) the symbolized leg movements in the sagittal plane are identical (i.e. \( b-c-d-a \)). This Section explores the possibility to differentiate between the three motions using the timing of when the symbol occurred as an additional information.

The motion data for this Section was obtained using three motion databases:

1. HDMcuts [123]: manually cut versions of the longer HDM motion capture database, grouped by motion class.
CHAPTER 5. MOTION FEATURE EXTRACTION

2. CMU [96]: Carnegie-Mellon University motion capture database.

3. UOW: uncleaned motion capture database captured using the Moven suit [91], recorded at the University of Wollongong, 2011.

An experiment was conducted by extracting the symbol sequence of three classes of motion: walking two steps starting with the left leg, running two steps starting with the left leg, and sneaking two steps starting with the left leg. The timestamp of occurrence of each symbol in the b-c-d-a sequence was noted and aligned to the time of occurrence of the first symbol (i.e. “b” in the b-c-d-a sequence) where in this experiment ‘timestamp’ is counted in 1/120 of a second. For example, if symbol “b” occurs in timestamp 10, “c” in timestamp 20, “d” in timestamp 40, and “a” in timestamp 70:

- Timestamp where b occurs: 10. Therefore $timestamp_b = 10 - 10 = 0$
- Timestamp where c occurs: 20. Therefore $timestamp_c = 20 - 10 = 10$
- Timestamp where d occurs: 40. Therefore $timestamp_d = 40 - 10 = 30$
- Timestamp where a occurs: 70. Therefore $timestamp_a = 70 - 10 = 60$

After timestamp alignment, the final recorded timestamp for the sequence is therefore (0,10,30,60) from the original (10,20,40,70) for each symbol in the sequence b-c-d-a, and this forms a single data point in the experiment after the timestamps are converted to their actual seconds value by multiplying each timestamp value by 1/120. Basic statistics (i.e. mean and 95% confidence interval) are then calculated for the aligned timestamp of each symbol to determine if each symbol occurs at a certain time in a class of motion that is differentiable to another class of motion.

**Walking Motion**

Starting with the walking motion, Figure 5.3.3 shows the aligned symbol timing result on symbol sequence of b-c-d-a from all three databases (walking motion is present in all three databases). The result are:

- For the HDMcuts database, symbol $c$ occurs at 0.2 ($\pm 0.01$) seconds after $b$, symbol $d$ occurs at 0.62 ($\pm 0.02$) seconds after $b$, and symbol $a$ occurs at 0.81 ($\pm 0.03$) seconds after $b$.
- For the UOW database, symbol $c$ occurs at 0.23 ($\pm 0.01$) seconds, symbol $d$ occurs at 0.65 ($\pm 0.01$) seconds, and symbol $a$ occurs at 0.9 ($\pm 0.02$) seconds.
- For the CMU database, symbol $c$ occurs at 0.16 ($\pm 0.003$) seconds, symbol $d$ occurs at 0.6 ($\pm 0.004$) seconds, and symbol $a$ occurs at 0.7 ($\pm 0.005$) seconds.
Figure 5.3.3: The timings of symbol occurrences in walking motion from three different databases. The symbol $b$ serves as the starting point of time measurements to the other symbols, hence the time that $b$ occurred is always zero.
CHAPTER 5. MOTION FEATURE EXTRACTION

The result for the walking motion shows that the CMU database contains a relatively quicker walking motion compared to the HDMcuts and the UOW database. The CMU database also contains a highly regular walking motion, with very tight dispersion (less than 0.005 second variation) in the time of occurrence of each symbol. The consistency of the symbol timings in the CMU database is due to the walking motion being performed by a single person who are recorded in multiple takes over a relatively small area (3m x 8m) [96]. Overall, the small variability in the symbol timings in walking motion (less than 0.05 of a second) shows that the timings of a walking motion across individuals are extremely regular.

Running Motion

For the running motion, Figure 5.3.4 shows the symbol timing result on symbol sequence of b-c-d-a from two databases (i.e. UOW and CMU as the HDMcuts database does not contain visually comparable running motions). The symbol timing results are:

- For the UOW database, symbol c occurs at 0.09 (±0.005) seconds, symbol d occurs at 0.4 (±0.007) seconds, and symbol a occurs at 0.5 (±0.01) seconds.

- For the CMU database, symbol c occurs at 0.06 (±0.004) seconds, symbol d occurs at 0.36 (±0.005) seconds, and symbol a occurs at 0.44 (±0.005) seconds.

Similar to the walking motion, the running motion in the CMU database shows a quicker running motion compared to the UOW database. Also, for both databases, there is little variability (less than 0.007 seconds) in the symbol occurrences. Therefore, similar to the walking motion, the symbol timings of the running motion are very regular.

Sneaking Motion

Figure 5.3.5 shows the symbol timing result on symbol sequence of b-c-d-a from the HDMcuts database, the only database having the sneaking motion. The sneaking motion can be viewed as a stealthy/slow walk designed to create as little noise as possible during the motion, hence it should be comparable to the walking motion (which is evident in the sagittal plane leg movement symbols exhibited in Table 5.3.1).

In the sneaking motion analyzed, symbol c occurs at 0.34 (±0.04) seconds, symbol d occurs at 0.93 (±0.07) seconds, and symbol a occurs at 1.22 (±0.1) seconds. The symbol timing analysis result shows that there is a relatively large variation on the symbol timings involved in the motion compared to, e.g., walking or running. However, the error bars which represent the 95% confidence intervals in Figure 5.3.5
Figure 5.3.4: Running motion analysis from two different databases. The symbol $b$ serves as the starting point of time measurements to the other symbols, hence the time that $b$ occurred is always zero.
Figure 5.3.5: Sneaking motion analysis. The symbol b serves as the starting point of time measurements to the other symbols, hence the time that b occurred is always zero.

do not show any overlap between the symbol timings, which shows that the symbol sequence of b-c-d-a is consistent.

Walking vs. Running vs. Sneaking Symbol Timing Comparisons

Grouping the timing results from walking (Figure 5.3.3), running (5.3.4), and sneaking (5.3.5) results in the overall symbol timing comparison of the three motions shown in Figure 5.3.6:

- Averaging across all walking motions tested, symbol c occurs at 0.2 (±0.007) seconds, symbol d occurs at 0.6 (±0.01) seconds, and symbol a occurs at 0.8 (±0.02) seconds.

- Across all running motions tested, symbol c occurs at 0.08 (±0.005) seconds, symbol d occurs at 0.4 (±0.008) seconds, and symbol a occurs at 0.49 (±0.01) seconds.

- In the sneaking motion analyzed, symbol c occurs at 0.34 (±0.04) seconds,
Figure 5.3.6: Average symbol timings between three types of motion that generates identical symbol sequences (b-c-d-a). The error bars which represent 95% confidence interval do not overlap between the three types of motions having the same symbol sequence, showing a statistical difference between the symbol timings of the three motions. Therefore, using the timing information in combination with the symbols, motions that generate identical symbol stream can still be differentiated from one another.

Symbol $d$ occurs at 0.93 (±0.07) seconds, and symbol $a$ occurs at 1.22 (±0.1) seconds.

The error bars in Figure 5.3.6 do not overlap between the symbol timings of each motion classes, and the variability of the symbol timings in walking and running motions are less than 0.02 seconds, which signifies that the timing of the symbol occurrences in walking and running are consistent across databases, the performer’s limb length, and walking/running pattern. Sneaking motion exhibits a relatively large variability in the symbol timings compared to walking and running, but the symbol occurrence timings do not vary by more than 0.1 seconds, hence the timings of a sneaking motion are still relatively regular.

### 5.3.4 Discussion

Section 5.3.2 has shown that the extracted symbol sequences have consistent timings even when the source motion capture data was not post-processed (i.e. cleaned up for jitters and errors in the motion sensors). Capturing the walking motion of four different subjects with large variation in height and large variation in walking patterns in a 5m x 5m room shows that the extracted symbol sequences for the sagittal movements of the legs are consistent with the walking motions extracted from other databases (e.g. HDM and CMU)
Although individual limb movements can be determined using the extracted symbolic representation, there are certain cases where different motions resulted in identical symbols being extracted when the movement of a certain limb is being considered. For example, walking, running, and sneaking converges into the same representation in the symbol sequence of $b-c-d-a$. However, the three motions can be differentiated by the timings of the symbol, which was demonstrated in Figure 5.3.6. In conclusion, to differentiate motions having the same symbol sequence, the timings of each symbol’s occurrence would need to be taken into account and therefore should be considered part of the extracted symbol sequence.

5.4 Symbol Sequence Use Cases and Applications

Having a representation of human motion in the form of the symbol stream leads to two possible applications in multimedia space: automatic motion annotation and matching. Subsection 5.4.1 explores how the symbol sequences and the symbol timings could be used to automatically annotate a motion capture recording into a semantic HMML representation and subsection 5.4.2 explores the use of symbol sequences and timings in a matching application.

5.4.1 Automatic Annotation of Motion Capture Data using HMML

HMML (in Chapter 4) was designed to be able to describe human motions from any source. While it is possible to manually create an HMML description of a human motion, having the capability to extract HMML descriptions automatically is desired due to the amount of user-generated content currently available on the Internet. As a preliminary step, this section presents a method to automatically extract HMML descriptions from motion capture data using the symbol sequences.

Since the symbol sequence extracted using the technique described in Section 5.2 also contains the timing information of when each symbol occurs in time, it is possible to convert the symbol sequences into its HMML representation automatically. The overall process of HMML annotation is illustrated in Figure 5.4.1. In Figure 5.4.1, Step 1 is the motion to be processed (motion capture data in BVH format in this case); Step 2 performed partial reconstruction as detailed in Section 5.2.2; Step 3 identifies the turning points of the reconstructed angular values as detailed in Section 5.2.3 (the arms and the legs in the sagittal plane in this case); finally Step 4 translates the turning points into their HMML counterpart.

Figure 5.4.2 shows the angular value plot along with the assigned symbols for leg movement in the sagittal plane. There are two sub-motions of interest in this
Figure 5.4.1: The overall process of HMML extraction from motion data.
walking motion:

1. The left leg stepping forward (defined as the symbol sequence of $a$-$b$-$c$).
2. The right leg stepping forward (defined as the symbol sequence of $c$-$d$-$a$).

For brevity, only the leg movements in the sagittal plane is used in the example in this Section. In Figure 5.4.2, there are four occurrences of the symbols of interest:

- Sequence $a$-$b$-$c$ which occurs from frame 55 (symbol $a$) to frame 132 (symbol $c$);
- Sequence $c$-$d$-$a$ which occurs from frame 132 (symbol $c$) to frame 207 (symbol $a$);
- The second $a$-$b$-$c$ sequence which occurs from frame 207 (symbol $a$) to frame 283 (symbol $c$); and finally
- The second $c$-$d$-$a$ sequence which occurs from frame 283 (symbol $c$) to frame 360 (symbol $a$).

Since both sequences $a$-$b$-$c$ and $c$-$d$-$a$ were defined as the left leg stepping forward and the right leg stepping forward respectively, the definition of the two sequences are correlated to the HMML schema of:

- \(<\text{leg side}="\text{left}\" \text{dir}="\text{forward}\">\) for the sequence $a$-$b$-$c$; and
- \(<\text{leg side}="\text{right}\" \text{dir}="\text{forward}\">\) for the sequence $c$-$d$-$a$.

The resulting HMML annotation is shown in Figure 5.4.3. A partial list (for brevity) showing the correlation between the symbol sequences and their HMML counterpart is shown in Table 5.4.1.

5.4.2 Motion Matching

As an initial step in creating the multimedia search using human motion as depicted in Figure 5.0.1, an experiment similar to the “uniqueness” test (performed in Section 5.2.5) is described in this section to determine the feasibility of using the symbol sequences to act as a feature vector for searching and matching human motion. The goal of the experiment is to determine if symbol sequences extracted at a low MDT value could be matched to symbols extracted at a higher MDT value and vice versa, the consequence of which is that the matching algorithm (i.e. the Echoprint [9] algorithm used in this work and discussed in 2) in combination with the symbol sequences are capable of finding a match if the input sequence is similar but not identical to the sequences that are present in the codebook. For example,
Figure 5.4.2: The source angular plot from which automatic HMML annotation is performed.

<?xml version="1.0" encoding="UTF-8"?>
<hmml
xmlns="urn:uow:hmml:schema:2012"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

<motion id="walk">
  <sagittal>
    <!-- left leg forward: sequence of a-b-c -->
    <leg side="left" dir="forward" start="55" end="132"/>

    <!-- right leg forward: sequence of c-d-a -->
    <leg side="right" dir="forward" start="132" end="207"/>

    <!-- left leg forward: sequence of a-b-c -->
    <leg side="left" dir="forward" start="207" end="283"/>

    <!-- right leg forward: sequence of c-d-a -->
    <leg side="right" dir="forward" start="283" end="360"/>
  </sagittal>
</motion>
</hmml>

Figure 5.4.3: An example of an HMML document derived for a walking motion. This example shows the left leg stepping forward from frames 55 to 132, the right leg stepping forward from frames 132 to 207, the left leg stepping forward from frames 207 to 283, and the right leg stepping forward from frames 283 to 360.
<table>
<thead>
<tr>
<th>Movement</th>
<th>Symbol sequence</th>
<th>HMML representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left leg stepping forward</td>
<td>a-b-c</td>
<td><code>&lt;leg side=&quot;left&quot; dir=&quot;forward&quot;&gt;</code></td>
</tr>
<tr>
<td>Right leg stepping forward</td>
<td>c-d-a</td>
<td><code>&lt;leg side=&quot;right&quot; dir=&quot;forward&quot;&gt;</code></td>
</tr>
<tr>
<td>Left leg stepping backward</td>
<td>b-a-d</td>
<td><code>&lt;leg side=&quot;left&quot; dir=&quot;backward&quot;&gt;</code></td>
</tr>
<tr>
<td>Right leg stepping backward</td>
<td>d-c-b</td>
<td><code>&lt;leg side=&quot;right&quot; dir=&quot;backward&quot;&gt;</code></td>
</tr>
<tr>
<td>Left leg stepping left</td>
<td>f-e</td>
<td><code>&lt;leg side=&quot;left&quot; dir=&quot;left&quot;&gt;</code></td>
</tr>
<tr>
<td>Left leg stepping right</td>
<td>e-f</td>
<td><code>&lt;leg side=&quot;left&quot; dir=&quot;right&quot;&gt;</code></td>
</tr>
<tr>
<td>Right leg stepping left</td>
<td>h-g</td>
<td><code>&lt;leg side=&quot;right&quot; dir=&quot;left&quot;&gt;</code></td>
</tr>
<tr>
<td>Right leg stepping right</td>
<td>g-h</td>
<td><code>&lt;leg side=&quot;right&quot; dir=&quot;right&quot;&gt;</code></td>
</tr>
<tr>
<td>Left arm moving forward</td>
<td>m-n</td>
<td><code>&lt;arm side=&quot;left&quot; dir=&quot;forward&quot;&gt;</code></td>
</tr>
<tr>
<td>Left arm moving backward</td>
<td>n-m</td>
<td><code>&lt;arm side=&quot;left&quot; dir=&quot;backward&quot;&gt;</code></td>
</tr>
<tr>
<td>Right arm moving forward</td>
<td>o-p</td>
<td><code>&lt;arm side=&quot;right&quot; dir=&quot;forward&quot;&gt;</code></td>
</tr>
<tr>
<td>Right arm moving backward</td>
<td>p-o</td>
<td><code>&lt;arm side=&quot;right&quot; dir=&quot;backward&quot;&gt;</code></td>
</tr>
</tbody>
</table>

Table 5.4.1: The partial translation list (for brevity) showing the correlation between the symbol sequences and its HMML description counterpart.
as previously shown in Figure 5.2.10, the time-arranged symbol sequence of a walking forward motion is \textit{c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c}. The goal is to match this string of symbols (the “\textit{input sequence}”) with a known “left leg steps forward” motion having the symbol sequence of \textit{a-b-c} (the “\textit{codebook sequence}”).

A simplified illustration of the Echoprint algorithm method to perform this uniqueness testing is shown in Figure 5.4.4. For example, consider the previous two symbol sequences: \textit{c-n-d-o-a-b-p-m-c-n-d-o-a-b-p-m-c} as the input sequence, and \textit{a-b-c} as the sequence to be the sequence to be compared to in the codebook. Writing the symbol sequence in the form of \textit{FrameNumber–Symbol} and applying it to both symbol sequences results in:

- Step 1: Input sequence vs. the codebook sequence
  
  \begin{itemize}
  \item 10 c 60 n 61 d 73 o 90 a 130 b 135 p 148 m 150 c 195 n 196 d 205 o 220
  \item a 260 b 262 p 270 m 280 c
  \end{itemize}

  for the input sequence, and;

  \begin{itemize}
  \item 0 a 40 b 60 c
  \end{itemize}

  for the codebook sequence (left leg steps forward).

  where the sequence \textit{10 c 60 n 61 d} means that the symbol \textit{c} occurs at frame 10, symbol \textit{n} occurs at frame 60, and symbol \textit{d} occurs at frame 61.

  Matching the input sequence to the codebook sequence can thus be achieved by extracting the relevant symbols along with the associated frame numbers and comparing the relative occurrence of the symbols:

  - Step 2: Identify the relevant frame-symbol pairs in the input sequence:
    
    Relevant \textit{frame–symbol} pairs that exist in the input sequence: \textit{90 a 130 b 150 c} and \textit{220 a 260 b 280 c}

  - Step 3: Symbol delay calculation:
    
    Timing between relevant symbols:
    
    \begin{itemize}
    \item In the first sequence, from \textit{a} to \textit{b}: 130-90=40 frames, from \textit{a} to \textit{c}: 150-90=60 frames
    \item In the second sequence, from \textit{a} to \textit{b}: 260-220=40 frames, from \textit{a} to \textit{c}: 280-220=60 frames
    \end{itemize}

  - Step 4: Compare the relevant frame–symbol pairs found in the input sequence to the codebook sequence:
    
    The delays are identical to the known motion of left leg steps forward:
CHAPTER 5. MOTION FEATURE EXTRACTION

From a to b: 40-0=40 frames, and from a to c: 60-0=60 frames

Therefore, the two occurrences of the motion “left leg steps forward” are recognized to exist in the input sequence.

To simplify the experiment and to enable the use of the existing Echoprint algorithm used for the previous uniqueness experiment, the tests were performed by exploiting the fact that symbols extracted at a low MDT value formed the super-set of symbols extracted at a higher MDT values (in Equation (5.2.5)). Therefore, symbol sequences extracted from a motion capture recording at MDT value of 0 (i.e. Motion1-MDT-0) should be identified as a match to symbol sequences from the same motion capture recording extracted at a higher MDT value (i.e. it should be identified as a match to Motion1-MDT-1, Motion1-MDT-5, Motion1-MDT-10, etc.).

As a secondary step, another experiment was also performed to determine if the reverse case is true; that is, does Motion1-MDT-10 match Motion1-MDT-0, which signifies that symbols extracted from a motion by using a high MDT value could match the symbols extracted from the same motion extracted using a lower MDT value.

Therefore, there are two experiments to be performed in this Section: The first part of the experiment simulates a high-detail motion database, i.e. the incoming sequence was extracted at an MDT value lower than the MDT value of the database. The second part of the experiment simulates a low-detail motion database, i.e. the incoming sequence was extracted at an MDT value higher than the MDT value of the database. Scoring for the experiment is using the previously utilized Precision (Equation (5.2.6)) and Recall (Equation (5.2.7)) values.

As in the uniqueness experiment, the same HDMcuts database was used in this experiments. The contents of the HDMcuts database was previously described in Section 5.2.5. MDT values used in both experiments are 0, 1, 5, 10, 15, and 20 similar to the MDT values of the uniqueness test due to the same reasoning (i.e. the error bars shown in the average number of symbols plot in Figure 5.2.13 do not overlap and hence there are statistical difference between symbols extracted at those MDT values).

Also similar to the uniqueness experiment in Section 5.2.5, four separate window sizes are used: 1 second; 2 seconds; 3 seconds; and 4 seconds. For every window size value, the incoming query was cropped according to the window size being tested. For example, using a 1 second window, only the first second of the input query is used for matching. Using a 2 seconds window, only the first 2 seconds of the input query is used, etc. Window sizes of up to 4 seconds are used due to the motion database’s average motion length, which is 3.1 seconds.
Figure 5.4.4: A method to match a known symbol sequence to an input symbol sequence for motion. This matching technique is used in the Echoprint music matching algorithm [9] and was utilized for this experiment. The steps involved are: (1) The input symbol sequence; (2) Identification of relevant frame-symbol pairs; (3) Calculation of the time differences between symbols; and (4) Comparison with a known sequence in the database.
High Detail Motion Database

For the high-detail motion database experiment, the motion database was extracted using MDT values of 0 (which is the lowest possible MDT value), and the input sequences are extracted using MDT values >0, i.e. values of 1, 5, 10, 15, and 20. The results of the experiment are shown in Figure 5.4.5.

Figure 5.4.5 shows that if the input sequence is a subset of the motion database (i.e. the database was extracted using a low MDT value and the input sequence was extracted at a higher MDT value), a database of symbol sequences extracted using an MDT value of 0 for 1 second of motion is able to be matched against any incoming sequences which is the subset of the database symbol sequence, and an MDT value of 5 resulted in precision score of 0.99 ±0.007 and a recall score of 1 ±0.004. For a motion of 1 second in length, higher MDT values of 10, 15, and 20 in the incoming sequence results in poor precision and recall scores by comparison, with none of them scoring above 0.4 in both precision and recall scores.

Increasing the window size results in a corresponding increase in the precision and recall scores of input sequences extracted at the higher MDT values: a window size of 2 seconds resulted in precision and recall scores at MDT values of 10 comparable to the scores obtained by MDT value of 1 and 5, and window sizes of 3 and 4 seconds in Figure 5.4.5 showing comparable results and relatively high precision and recall scores for all MDT values.

The observation of the results of the high-detail motion database experiment reveals that the trend of the precision and recall scores in Figure 5.4.5 is very similar to the uniqueness experiment results shown in Figure 5.2.14, where the uniqueness experiment results show the ideal conditions where both the input sequence and the database both extracted at the same MDT value. Therefore, in a matching application, it is desirable to have the incoming sequence extracted at a lower MDT value relative to the MDT value of the database.

Low Detail Motion Database

For the low-detail motion database experiment, the motion database was extracted using MDT value of 20, and the input sequences are extracted using MDT values <20, i.e. values of 0, 1, 5, 10, and 15. The results of the experiment is shown in Figure 5.4.6.

Figure 5.4.6 shows that if the input sequence is a superset of the motion database (i.e. the database was extracted using a high MDT value and the input sequence was extracted at a lower MDT value), similar pattern of relatively high precision and recall scores are observed as in the high-detail database (Figure 5.4.5) and the uniqueness experiment (Figure 5.2.14) but with lower overall precision and recall.
Figure 5.4.5: Plots showing the precision and recall values of the high-detail motion database experiment using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x-axis denotes the MDT value used in the extraction process for the input sequence, and the motion database was extracted using an MDT value of 0. Error bars represent 95% confidence intervals.
Figure 5.4.6: Plots showing the precision and recall values of the low-detail motion database experiment using four different window sizes: 1 second, 2 seconds, 3 seconds, and 4 seconds. The x axis denotes the MDT value used in the extraction process for the input sequence, and the motion database was extracted using an MDT value of 20. Error bars represent 95% confidence intervals.
scores. For example, incoming sequences extracted using an MDT value of 5 (for 1 second of motion) resulted in precision score of $0.65 \pm 0.03$ and recall score of $0.76 \pm 0.03$ (which are lower than precision score of $0.99 \pm 0.007$ and a recall score of $1 \pm 0.004$ in the high-detail database experiment in Figure 5.4.5). The worst result in this experiment is when the incoming sequence are extracted using MDT value of 15 using only 1 second of motion, having a precision score of $0.16 \pm 0.02$ and a recall score of $0.18 \pm 0.03$. However, increasing the window length of the motion to be matched shows improvements in the higher MDT values, such as 10 and 15.

The low-detail database experiment therefore shows that although it is possible to perform matching using a low-detail database vs. a higher-detailed input sequence, it is not desirable since it results in lower overall precision and recall scores for all MDT values tested.

**Discussion**

From the results in Figures 5.2.14, 5.4.5, and 5.4.6, it was shown that varying the MDT values in the database and in the input sequence show the same general trend:

- Using one second of a motion, incoming sequences extracted using MDT values of 0, 1, and 5 achieved relatively high precision and recall scores, and MDT values above 5 shows poor precision and recall scores.

- Using two seconds of a motion, precision and recall scores for incoming sequences extracted using MDT value of 10 improves to be comparable to the lower (i.e. 0, 1, and 5) MDT values.

- Using motions of three and four seconds in length, all MDT values produced comparable precision and recall scores.

Hence in the three cases tested (uniqueness, high-detail database, and low-detail database), MDT values of 5 for the incoming sequence should be considered as the minimum MDT value to be used in a matching scenario, as it provides a relatively stable performance. Also, the best precision and recall scores are achieved if the database is either extracted using the lowest thresholds possible; or extracted using the same threshold as the incoming query.

In terms of the length of motion required to differentiate between motions, the results show that the longer the motion, the better the chance of a correct match. From the results in Figure 5.2.14, 5.4.5, and 5.4.6, even extracting both the codebook and the query using MDT value of 20, a correct match can still be found if enough motion duration is used in the matching process (i.e. 3 seconds of motion or more). However, if only short motions are available (e.g. only up to 1 second in length), then using MDT values of 5 for both the database and the input sequence provides
competitive results to that of extracting at an MDT value of 0, with the advantage of having only 40% of the number of symbols while having a comparable matching performance.

5.5 Conclusions

This Chapter has presented a method to extract features from motion capture data in the form of a symbol sequence. The partial reconstruction technique was introduced to allow the symbols to be comparable from one motion to another motion due to the manipulation in the origin point of motion capture data reconstruction. For the walking and running motions, the extracted symbols are shown to be robust and produced comparable symbol sequence even when there are large variations in the person’s height and movement pattern.

The MDT value for the extraction process was also introduced to allow for a post-processing stage in the symbol extraction process to remove movements considered to be jitter or errors in the source motion capture recording. As an additional advantage, the MDT values could act as a scaling mechanism to allow more or less movements to be detected during the symbol extraction process. Experiments performed in this chapter show that extraction using an MDT value of 0 preserves enough detail of the uniqueness of a motion recording, while MDT values higher than 0 provides savings in the space required to store the symbols in a database.

Further, an analysis comparing motions with similar sub-motion (i.e. the leg-forward sub-motion in walking, running, and sneaking) shows that to differentiate between the three motions, the information of when each symbol occur in time would have to be considered as part of the symbol sequence.

Section 5.3 provided an analysis of motion capture data from three different motion databases (UOW, CMU, and HDMcuts) and found that during walking, running, and sneaking, the timing differences for each motion class are very small so that the three motions are differentiable from one another. Notably, the UOW database captured the motions of four individuals with radically different anthropometry measurements (section 5.3.2 tests subjects with height of 167 cm, 190 cm, 185 cm, and 147 cm) and found that walking motions are relatively consistent for the four individuals, even with the walking motion having different patterns. This supports the conclusion by Weyand et al. [126], that people with different anthropometry (i.e. static) measurement move in the same manner.

Two possible applications using the symbol sequences are discussed: automatic metadata extraction from motion capture data (using HMML as the metadata format) and a motion matching application (for duplicate detection or determining the similarities between one motion to another motion) was presented. In a matching
application, the symbol sequence was demonstrated to be able to work in a system utilizing an existing audio matching technique of Echoprint. Experiments using the Echoprint algorithm in cases where the symbol sequence database contains symbols extracted using a low MDT value (high-detail database) and symbols extracted using a high MDT value (low-detail database) show that symbol extraction using an MDT value of 5 preserves enough unique detail in each motion recording while simultaneously extracting 40% less symbols compared to symbol extraction using an MDT value of 0.

Ultimately, the goal is to be able to extract human motion metadata from any source material, both in 2D and 3D. There is ongoing work investigating conversion from 2D motion to 3D (such as VideoMocap [64]), and if such technologies become more commonplace in the future, the partial reconstruction technique presented in this chapter could potentially be used to extract motion descriptions from a 2D video. Since the techniques described in this chapter provide a method to extract information from 3D motion data, future work in this area should consider the 2D-to-3D conversion performed by e.g. VideoMocap as a starting point of extracting movement information from a 2D video.
Chapter 6

Conclusions and Future Work

This Chapter presents the conclusions of Chapters 3, 4, and 5. This Chapter also presents possible directions for future work in the area.

6.1 Conclusions

Survey conducted on 3 million Youtube videos [8] revealed that 30% of the video on Youtube are classified as “sport” and/or “music”, which contain human motion. Although standard multimedia description schemes that objectively described the media content exist in the form of Dublin Core and MPEG-7, both standards do not specify how to describe human motion objectively. Also, no common query format exist in the multimedia space that is capable to perform multimedia query on multiple databases described using different multimedia description standards. Recognizing this shortcoming, MPEG created the call for proposal for the upcoming MPEG-7 Query Format (MP7QF) standard, which will standardize how multimedia databases described using MPEG-7 should be queried.

Multimedia Query Format (MQF) presented in Chapter 3 was one of the proposals submitted to MPEG during the MP7QF standardization effort, and took part in many discussions and suggestions to improve the query format specifications and requirements. Further, the validity of MQF to perform multimedia queries are validated by trials and experiments organized by MPEG, where the RPN approach of MQF and its ability to include other multimedia description standard (such as Dublin Core) in a single query were extensively discussed and investigated by the participants of the MP7QF standardization effort.

To achieve the goal of human motion multimedia query, a new searchable human motion description was also needed. To address this need, the new Human Motion Markup Language (HMML) was presented in Chapter 4. HMML is an XML-based human motion description format that is easy to incorporate into multimedia queries using MQF, since HMML was not designed to replace any existing
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

multimedia description standard. Instead, HMML was designed as an additional query term involving human motion that works in conjunction with existing description formats such as MPEG-7 and Dublin Core to describe the desired media in more detail compared to using only a single multimedia description scheme. Key aspect of HMML include its search-centric design, motion description methodologies that was inspired by the popular Laban dance notation, and human-readable XML representation of motion to ease description creation and processing using the abundance of XML tools available.

Automatic extraction of HMML from a media is also one of the principal goal of this thesis, and Chapter 5 presented a method to extract HMML description automatically from motion capture data. Chapter 5 also validates the consistency of human motion using features (i.e. symbols) extracted from motion capture data by analyzing the similarities in timings of walking and running motions performed by people of different heights and movement patterns. Analysis of the effect of the Motion Detection Threshold (MDT) values used during the symbol sequence extraction process was also performed in Chapter 5. Furthermore, Chapter 5 also presented a motion matching method using the symbol sequences extracted from motion capture data via an existing audio matching algorithm (Echoprint [9]) with experiments to determine the optimal parameters of the feature extraction process.

6.2 Future Works

- **HMML compatibility with existing multimedia description formats.**
  
The fact that human motion search is not an established field argues for interoperability with existing description schemes. HMML is, and will be, a separate entity from any existing multimedia description scheme (e.g. MPEG-7 or Dublin Core), and was not designed exclusively for use with either MPEG-7 or Dublin Core. Instead, HMML needs to be an additional aspect of a multimedia description, aside from conventional metadata such as author, title, color description, etc.

  HMML description would need to serve as a bridge between highly temporal and non-temporal multimedia description formats for search purposes, and the motion description should be used in conjunction with existing metadata descriptions to perform human motion centric multimedia search.

- **Automatic 2D video to HMML extraction.**
  
  Ultimately, the goal is to be able to extract HMML from any source material, both in 2D and 3D. There are ongoing work investigating conversion from 2D motion to 3D (such as VideoMocap [64]), and if such technologies become more
commonplace in the future, the partial reconstruction technique presented Chapter 5 could potentially be used to extract HMML description from 2D video, enabling automatic HMML extraction from 2D sources.

• **Scoring system to improve motion search.**

Chapter 5 has discussed the possibility of using an existing algorithm (i.e. Echoprint [9]) to perform motion matching using the symbol sequences. However, the Echoprint algorithm was designed to find an exact match of an incoming query to a database. A more flexible algorithm could be developed to take into account similarities between the input query and the motion database and assign a similarity score to each entry in the database. These similarity scores can subsequently be used to rank the result set of a search. For example, if the input query is a walking motion, a score of 100 to a particular database entry signifies that the input query is an exact match of that particular entry. A score of 90 could signify that the database entry is similar to walking, but with some minor differences (e.g. a longer step, an exaggerated side-to-side movements of the legs, etc.), while a score of 0 signifies that the database entry has nothing in common with the input query. This ranked result set would give additional information to the user regarding how similar a motion is to another, and a threshold of a similarity score could be used to determine if a motion is visually similar or dissimilar, allowing a more informed decision for the user to find the desired motion.

• **Motion classification using the symbol sequences.**

Chapter 5 has shown that motion matching is possible using the symbol sequences extracted from motion capture data, along with the timing information of the symbols. An improvement to the matching database to enable motion classification applications can be achieved by averaging the timing information of the symbol sequences and populate the motion database with the averaged symbol sequences instead of individual motion instances. For example, instead of the database containing (in frame number – symbol format):

- motion "walk 1": 0 a 10 b 20 c 30 d
- motion "walk 2": 6 a 16 b 26 c 36 d

The database could be populated with the averaged timing information instead, hence:

- motion class “walk”: 3 a 13 b 23 c 33 d
and thus perform motion classification by comparing the input query to the motion classes in the database. The overall system that uses a combination of motion timing averages and the similarity scoring system will create a motion classification which will result in similarity scores between an input query and all the motion classes in the database, allowing the user to determine which motion class is most similar/dissimilar to the query. The use of a similarity score in this case would also prevent the classification system to classify an input query to an erroneous class (e.g. classifying jumping jack into running if the database does not have the jumping jack motion class) and to allow the system to fail gracefully if the input motion is extremely dissimilar to the database contents (e.g. by returning low similarity scores).
Bibliography


[16] “MPEG-7 overview.”

[18] “ID3.org.”


[22] “Frames - ID3.org.”


[29] W3C, “XSL transformations (XSLT).”


[33] W3C, “RSS 2.0 specification.”

[34] IETF, “RFC 4287 - the atom syndication format.”

[35] “World wide web consortium (W3C).”
[36] “Internet engineering task force (IETF).”

[37] W3C, “XHTML 1.0: The extensible HyperText markup language (second edition).”

[38] IETF, “Extensible messaging and presence protocol (XMPP): core.”


[40] Facebook, “Facebook chat API - facebook developers.”


[47] Internet Assigned Numbers Authority (IANA), “Uniform resource names (URN) namespaces.”


[50] “Using dublin core.”

[51] “Expressing dublin core metadata using the DC-Text format.”

[52] “Expressing dublin core metadata using HTML/XHTML meta and link elements.”

[53] “Expressing dublin core description sets using XML (DC-DS-XML).”

[54] “Expressing dublin core metadata using the resource description framework (RDF).”


[61] “Google images.”


[73] “Biomotion lab.”


[90] W3Schools, “DTD tutorial.”

[92] “Motion capture systems from vicon.”


[105] N. S. Borenstein and N. Freed, “Multipurpose internet mail extensions (MIME) part two: Media types.”
[106] Internet Assigned Numbers Authority (IANA), “Hypertext transfer protocol (HTTP) status code registry.”


Appendix A

Multimedia Query Format XML Schema

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
    xmlns="urn:mpeg:mpeg7:mqf:schema:2007"
    targetNamespace="urn:mpeg:mpeg7:mqf:schema:2007"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
        schemaLocation="xml-1998.xsd"/>
    <xs:import namespace="urn:mpeg:mpeg7:schema:2001"
        schemaLocation="mpeg7-v1.xsd"/>
    <xs:import namespace="urn:uow:hmml:schema:2012"
        schemaLocation="HMML.xsd"/>
    <xs:import namespace="http://purl.org/dc/elements/1.1/"
        schemaLocation="dc.xsd"/>
    <xs:element name="mqf">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="queryId" type="xs:string"/>
                <xs:element name="from" type="xs:anyURI"/>
                <xs:choice>
                    <xs:element name="query" type="queryType"/>
                    <xs:element name="replies" type="repliesType"/>
                    <xs:element name="exceptions" type="exceptionsType"/>
                    <xs:element name="services" type="servicesType"/>
                </xs:choice>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
<xs:attribute ref="xml:lang"/>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
<xs:complexType name="queryType">
<xs:sequence>
<xs:element name="replyType" type="mpeg7:mimeType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="replyAs" type="xs:string"

<xs:element name="startIndex" type="xs:positiveInteger" minOccurs="0" />
<xs:element name="numOfResult" type="xs:positiveInteger" minOccurs="0" />
<xs:element name="sortBy" type="sortByType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="groupBy" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="timeOut" type="xs:positiveInteger" minOccurs="0" />
<xs:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="item" type="itemType" />
  <xs:element name="operator" />
</xs:choice>
</xs:sequence>
<xs:attribute name="stream" type="xs:boolean" />
<xs:attribute name="streamId" type="xs:string" />
</xs:complexType>
<xs:complexType name="repliesType">
<xs:choice>
  <xs:element name="text" type="xs:string" />
  <xs:element name="reply" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="item" type="mpeg7:MediaLocatorType" minOccurs="0" />
        <xs:element name="description" type="descriptionType" minOccurs="0" />
        <xs:element name="index" type="xs:positiveInteger" />
      </xs:sequence>
      <xs:attribute name="relevance" type="relevanceType" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:choice>
</xs:complexType>
<xs:complexType name="exceptionsType">
<xs:sequence>
  <xs:element name="exception" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="item" type="mpeg7:MediaLocatorType" minOccurs="0" />
        <xs:element name="description" type="descriptionType" minOccurs="0" />
        <xs:element name="index" type="xs:positiveInteger" />
      </xs:sequence>
      <xs:attribute name="relevance" type="relevanceType" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
APPENDIX A. MULTIMEDIA QUERY FORMAT XML SCHEMA

<x:schema>
<x:element name="code"/>
<x:element name="description" type="mpeg7:TextualType"/>
</x:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<x:complexType name="servicesType">
<x:element name="version" type="xs:string" minOccurs="0"/>
<x:element name="queryLevel" type="queryLevelType" minOccurs="0" maxOccurs="unbounded"/>
<x:element name="replyType" type="mpeg7:mimeType" minOccurs="0" maxOccurs="unbounded"/>
<x:element maxOccurs="unbounded" minOccurs="0" name="descriptionNamespace" type="descriptionNamespaceType"/>
<x:element name="operator" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
</x:sequence>
</xs:complexType>
</xs:complexType>
<x:complexType name="formatType">
<x:simpleContent>
<x:extension base="mpeg7:mimeType">
<x:attribute name="delimiter" type="xs:string"/>
<x:attribute name="ref" type="xs:anyURI"/>
</x:extension>
</xs:simpleContent>
</xs:complexType>
<x:simpleType name="relevanceType">
<x:restriction base="xs:integer">
<x:minInclusive value="0"/>
<x:maxInclusive value="100"/>
</x:restriction>
</xs:simpleType>
</xs:complexType>
<x:complexType name="sortByType">
<x:simpleContent>
<x:extension base="xs:string">
<x:attribute name="order" type="orderType"/>
</x:extension>
</xs:simpleContent>
</xs:complexType>
</xs:complexType>
<xs:simpleType name="orderType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ascending"/>
    <xs:enumeration value="descending"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="descriptionNamespaceType">
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
</xs:schema>
Appendix B

Human Motion Markup Language

XML Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:uow:hmml:schema:2012"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:uow:hmml:schema:2012"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

    <xs:simpleType name="sideType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="left"/>
            <xs:enumeration value="right"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="sagittalDirectionType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="forward"/>
            <xs:enumeration value="backward"/>
            <xs:enumeration value="center"/>
            <xs:enumeration value="none"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="sagittalLimbType">
        <xs:attribute name="side" type="sideType"/>
        <xs:attribute name="dir" type="sagittalDirectionType"/>
        <xs:attribute name="start" type="xs:positiveInteger"/>
    </xs:complexType>
</xs:schema>
```
<xs:attribute name="end" type="xs:positiveInteger"/>
<xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="sagittalPlaneType">
<x:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="arm" type="sagittalLimbType"/>
  <xs:element name="leg" type="sagittalLimbType"/>
  <xs:element name="torso" type="sagittalLimbType"/>
  <xs:element name="head" type="sagittalLimbType"/>
</xs:choice>
</xs:complexType>

<xs:simpleType name="coronalDirectionType">
<x:restriction base="xs:string">
  <xs:enumeration value="left"/>
  <xs:enumeration value="right"/>
  <xs:enumeration value="center"/>
  <xs:enumeration value="none"/>
</xs:restriction>
</xs:simpleType>

<xs:complexType name="coronalLimbType">
<xs:attribute name="side" type="sideType"/>
<xs:attribute name="dir" type="coronalDirectionType"/>
<xs:attribute name="start" type="xs:positiveInteger"/>
<xs:attribute name="end" type="xs:positiveInteger"/>
<xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="coronalPlaneType">
<x:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="arm" type="coronalLimbType"/>
  <xs:element name="leg" type="coronalLimbType"/>
  <xs:element name="torso" type="coronalLimbType"/>
  <xs:element name="head" type="coronalLimbType"/>
</xs:choice>
</xs:complexType>

<xs:simpleType name="transverseDirectionType">
<x:restriction base="xs:string">
</xs:restriction>
</xs:simpleType>
<xs:enumeration value="down"/>
<xs:enumeration value="up"/>
<xs:enumeration value="level"/>
<xs:enumeration value="none"/>
</xs:restriction>
</xs:simpleType>

<xs:complexType name="transverseLimbType">
  <xs:attribute name="side" type="sideType"/>
  <xs:attribute name="dir" type="transverseDirectionType"/>
  <xs:attribute name="start" type="xs:positiveInteger"/>
  <xs:attribute name="end" type="xs:positiveInteger"/>
  <xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="transversePlaneType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="arm" type="transverseLimbType"/>
    <xs:element name="leg" type="transverseLimbType"/>
    <xs:element name="torso" type="transverseLimbType"/>
    <xs:element name="head" type="transverseLimbType"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="motionDescriptionType">
  <xs:annotation>
    <xs:documentation>Grouped motions type definition</xs:documentation>
  </xs:annotation>
  <xs:choice maxOccurs="unbounded" minOccurs="0">
    <xs:element name="sagittal" type="sagittalPlaneType" minOccurs="0"/>
    <xs:element name="coronal" type="coronalPlaneType" minOccurs="0"/>
    <xs:element name="transverse" type="transversePlaneType" minOccurs="0"/>
    <xs:element name="motion" type="motionDescriptionType"/>
    <xs:element name="repeat" type="motionDescriptionRepType"/>
  </xs:choice>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="start" type="xs:positiveInteger"/>
</xs:complexType>
<xs:attribute name="end" type="xs:positiveInteger"/>
<xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="sagittalLimbRepType">
  <xs:attribute name="side" type="sideType"/>
  <xs:attribute name="dir" type="sagittalDirectionType"/>
  <xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="sagittalPlaneRepType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="arm" type="sagittalLimbRepType"/>
    <xs:element name="leg" type="sagittalLimbRepType"/>
    <xs:element name="torso" type="sagittalLimbRepType"/>
    <xs:element name="head" type="sagittalLimbRepType"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="coronalLimbRepType">
  <xs:attribute name="side" type="sideType"/>
  <xs:attribute name="dir" type="coronalDirectionType"/>
  <xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="coronalPlaneRepType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="arm" type="coronalLimbRepType"/>
    <xs:element name="leg" type="coronalLimbRepType"/>
    <xs:element name="torso" type="coronalLimbRepType"/>
    <xs:element name="head" type="coronalLimbRepType"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="transverseLimbRepType">
  <xs:attribute name="side" type="sideType"/>
  <xs:attribute name="dir" type="transverseDirectionType"/>
  <xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>

<xs:complexType name="transversePlaneRepType">
  <xs:attribute name="end" type="xs:positiveInteger"/>
  <xs:attribute name="duration" type="xs:positiveInteger"/>
</xs:complexType>
<xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="arm" type="transverseLimbRepType"/>
    <xs:element name="leg" type="transverseLimbRepType"/>
    <xs:element name="torso" type="transverseLimbRepType"/>
    <xs:element name="head" type="transverseLimbRepType"/>
</xs:choice>
</xs:complexType>

<xs:complexType name="motionDescriptionRepType">
    <xs:annotation>
        <xs:documentation>Repeated motions type definition</xs:documentation>
    </xs:annotation>
    <xs:choice maxOccurs="unbounded" minOccurs="0">
        <xs:element name="sagittal" type="sagittalPlaneRepType" minOccurs="0"/>
        <xs:element name="coronal" type="coronalPlaneRepType" minOccurs="0"/>
        <xs:element name="transverse" type="transversePlaneRepType" minOccurs="0"/>
        <xs:element name="motion" type="motionDescriptionType"/>
        <xs:element name="repeat" type="motionDescriptionRepType"/>
    </xs:choice>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="times" type="repeatTimesType"/>
</xs:complexType>

<xs:simpleType name="repeatTimesType">
    <xs:union memberTypes="xs:positiveInteger">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="unbounded"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:union>
</xs:simpleType>

<xs:complexType name="hmmlType">
    <xs:annotation>
        <xs:documentation>The HMML type definition</xs:documentation>
    </xs:annotation>
</xs:complexType>
<xs:choice maxOccurs="unbounded">
    <xs:element name="motion" type="motionDescriptionType">
        <xs:annotation>
            <xs:documentation>Grouped motions</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="repeat" type="motionDescriptionRepType">
        <xs:annotation>
            <xs:documentation>Repeated motions</xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:choice>

<xs:attribute name="fps" type="xs:positiveInteger"/>
<xs:attribute name="version" type="xs:string"/>
</xs:complexType>

<xs:element name="hmml" type="hmmlType">
    <xs:annotation>
        <xs:documentation>Main HMML document element</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:schema>
Appendix C

The Echoprint Algorithm

1. **Calculate the initial score based on features that are present in both the input and each entry in the codebook**

   The initial score is calculated using the number of features that are present in both the input query and in each entry in the codebook. For example in Figure C.0.1, the input query is compared against each entry in the codebook. In Figure C.0.1, codebook “Entry 1” initial score is 2 (since feature “1024810” and “1027459” both exist in the input query and the codebook entry 1), and codebook “Entry 2” initial score is 1 (since only feature “1024810” is present in both the input query and the codebook entry 2).

2. **Calculate the delay between features**

   For each matching features in the input query and the codebook, the timestamp differences are calculated. In the example shown in Figure C.0.2, in Entry 1, the delay for both features “1024810” and “1027459” are 10. In Entry 2, the delay for feature “1024810” is -20.

3. **Calculate the final score based on the delays**

   After the delays for each features are calculated, identical values of delays are grouped together and the number that exact number occurred is calculated as the final score. An example of final score calculation is shown in Figure C.0.3, where for each delay value calculated from Figure C.0.2, identical occurrence of a delay value are counted. In this example, Entry 1 has two counts of delays of 10, which results in a final score of 2 (two counts of delays of 10). For Entry 2, there is only one delay value (which is -20), therefore the final score for Entry 2 is 1 (one count of delay of -20).

4. **Make a decision for a match using both the initial and the final score**

   The Echoprint algorithm then makes a decision if the input query existed in the
Figure C.0.1: Echoprint algorithm initial scoring: The initial score is calculated for each entry in the codebook based on the features that are present in both the input query and the codebook entries. In this example, codebook “Entry 1” initial score is 2, and codebook “Entry 2” initial score is 1.

Figure C.0.2: Echoprint algorithm delay calculation: For each matching features in the input query and the codebook, the timestamp differences are calculated. In this example, in Entry 1, the delay for both features “1024810” and “1027459” are 10. In Entry 2, the delay for feature “1024810” is -20.
codebook by using both the initial and final scores, and calculates a weighting value that insures that the top score is sufficiently different compared to the second highest score. The criteria required to produce a match are:

(a)  \[ score_{top} \geq length_{fingerprint} \times 0.05 \]  \hspace{1cm} (C.0.1)

(b)  \[ score_{top} > \frac{score_{initial}}{4} \]  \hspace{1cm} (C.0.2)

(c)  \[ score_{top} - score_{second} \geq \frac{score_{top}}{3} \]  \hspace{1cm} (C.0.3)

where (C.0.1) ensures that the top-scoring match has enough fingerprint length, (C.0.2) ensures that the top score is also having a high initial score, and (C.0.3) ensures that the top score is sufficiently different from the second highest score. When all three criteria are satisfied, the algorithm then declares that a match is found.
Appendix D

The BVH Format

To reconstruct the full body, the transformation matrix of each bone would have to be calculated first. Once the transformation matrix of all bones are calculated, the full body can then be reconstructed by following the prescribed hierarchy of the skeleton in the motion capture file.

However, since so far only the movements of individual bones are reconstructed, one more thing that needs to be taken into account is the movement of the whole body. This is achieved by using a separate rotation and translation of a special bone designated as the “root” bone. Typically in motion capture, the root bone is located in the hip.

Figure D.0.1 shows an example of a fragment of a BVH file, where it shows that BVH format consists of two parts: the Hierarchy section, and the Motion section. For brevity, the bones shown in the example in Figure D.0.1 (and subsequently, Figures D.0.2-D.0.4) only shows the root, left hip joint, and left femur, hence the reconstruction example provided in this Section will reconstruct the movement of those three bones. The examples are taken from the file 35_01.bvh (available from the Carnegie-Mellon motion capture database [96]).

The Hierarchy section defines the skeleton structure described in a hierarchy of bones (similar to the hierarchy shown in Figure 2.4.12). The example shown in Figure D.0.1 shows three bones in the hierarchy, namely the hip (defined as ROOT, shown as \textit{hip}), the left hip joint (shown as \textit{lhipjoint}) and the left femur (shown as \textit{lfemur}).

Each bone defined in the hierarchy contains two values: the \textit{offset} and \textit{channels}. \textit{Offset} is the location of the bone’s local space with respect to the global space (where from a hierarchical point of view, the global space is the bone preceding the current one in the hierarchy). For example, the \textit{offset} values of \textit{lfemur} is shown as $1.77779 -1.88989 0.48267$, which means that the origin of the left femur bone is located at $x=1.77779$, $y=-1.88989$, $z=0.48267$ from the previous bone (in this case, \textit{lhipjoint}). Using the example in Figure 2.4.13, $P=[1.77779 -1.88989 0.48267]$. 


218
Figure D.0.1: An example of a BVH file fragment showing the hierarchy and motion of the hip, left hip joint, and left femur bones for one frame from the file 35_01.bvh (available from [96]).

Channels in the Hierarchy section describes the order of rotation, which governs the order of matrix multiplication that needs to be performed in Equation 2.4.18. In this example, the order is shown as Zrotation Yrotation Xrotation. Taking Equation 2.4.18 and letting \( X = \alpha, Y = \beta, \) and \( Z = \gamma \):

\[
R_{BVH} = R_{\gamma} \cdot R_{\beta} \cdot R_{\alpha} \tag{D.0.1}
\]

In the Motion section, the rotation of the bones are described in one frame per line (Note that the example shown in Figure D.0.1 shows only one frame). Each frame is described in terms of the angular value of the rotation of the bones according to the order described in the Hierarchy section. The first three numbers in the motion section is unique, in that they described the Xposition Yposition and Zposition of the root bone. The next set of three numbers described the rotations of the root bone (as shown in Figure D.0.2). Hence, the next set of three numbers describe the rotations of the left hip joint (as shown in Figure D.0.3), the next set of three numbers described the rotations of the left femur (Figure D.0.4) etc.

Reconstructing Motion Capture Data from BVH

Using the examples shown in Figure D.0.1, the motion reconstruction is as follows:

1. Construct the transformation matrices for each bone.
APPENDIX D. THE BVH FORMAT

HIERARCHY
ROOT hip {
  OFFSET 0.00000 0.00000 0.00000
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT lhipjoint {
    OFFSET 0 0 0
    CHANNELS 3 Zrotation Yrotation Xrotation
    JOINT lfemur {
      OFFSET 1.77779 -1.88989 0.48267
      CHANNELS 3 Zrotation Yrotation Xrotation
      ...
    }
  }
}
MOTION
Frames: 358 Frame Time: 0.033333
4.4005 17.8934 -21.0986 -7.4261 -8.3796 -0.9440 0.0000 0.0000 0.0000
-7.1426 -13.4851 -25.1391 ...

Figure D.0.2: BVH example: the root bone as defined in the skeleton hierarchy and its corresponding motion for the first frame.

HIERARCHY
ROOT hip {
  OFFSET 0.00000 0.00000 0.00000
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT lhipjoint {
    OFFSET 10 0 0
    CHANNELS 3 Zrotation Yrotation Xrotation
    JOINT lfemur {
      OFFSET 1.77779 -1.88989 0.48267
      CHANNELS 3 Zrotation Yrotation Xrotation
      ...
    }
  }
}
MOTION
Frames: 358 Frame Time: 0.033333
4.4005 17.8934 -21.0986 -7.4261 -8.3796 -0.9440 0.0000 0.0000 0.0000
-7.1426 -13.4851 -25.1391 ...

Figure D.0.3: BVH example: the left hip joint and its corresponding motion for the first frame.
APPENDIX D. THE BVH FORMAT

HIERARCHY
ROOT hip {
OFFSET 0.00000 0.00000 0.00000
CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
JOINT lhipjoint {
OFFSET 0 0 0
CHANNELS 3 Zrotation Yrotation Xrotation
JOINT lfemur {
OFFSET 1.77779 -1.88989 0.48267
CHANNELS 3 Zrotation Yrotation Xrotation
...
}
}
}
MOTION
Frames: 358 Frame Time: 0.033333
4.4005 17.8934 -21.0986 -7.4261 -8.3796 -0.9440 0.0000 0.0000 0.0000
-7.1426 -13.4851 -25.1391 ...

Figure D.0.4: BVH example: the left femur and its corresponding motion for the first frame.

- For the root bone as shown in Figure D.0.2, the translation and rotation values are:
  
  (a) 4.4005 17.8934 -21.0986 for the translation, and
  
  (b) -7.4261 -8.3796 -0.9440 for the rotation (described in the order of \( \gamma, \beta, \) and \( \alpha \)).

The rotation matrix for the root bone is therefore:

\[
R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-0.9440) & -\sin(-0.9440) \\ 0 & \sin(-0.9440) & \cos(-0.9440) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9999 & 0.0165 \\ 0 & -0.0165 & 0.9999 \end{bmatrix}
\]

\[
R_\beta = \begin{bmatrix} \cos(-8.3796) & 0 & \sin(-8.3796) \\ 0 & 1 & 0 \\ -\sin(-8.3796) & 0 & \cos(-8.3796) \end{bmatrix} = \begin{bmatrix} 0.9893 & 0 & -0.1457 \\ 0 & 1 & 0 \\ 0.1457 & 0 & 0.9893 \end{bmatrix}
\]

\[
R_\gamma = \begin{bmatrix} \cos(-7.4261) & -\sin(-7.4261) & 0 \\ \sin(-7.4261) & \cos(-7.4261) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.9916 & 0.1292 & 0 \\ -0.1292 & 0.9916 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
R_{\text{root}} = R_\gamma \cdot R_\beta \cdot R_\alpha = \begin{bmatrix} 0.9810 & 0.1316 & -0.1424 \\ -0.1279 & 0.9912 & 0.0352 \\ 0.1457 & -0.0163 & 0.9892 \end{bmatrix}
\]

By combining the translation values with the rotation matrix in a 4x4 matrix (as per Equation 2.4.15), the transformation matrix for the root bone is therefore:
APPENDIX D. THE BVH FORMAT

\[ T_{\text{root}} = \begin{bmatrix}
0.9810 & 0.1316 & -0.1424 & 4.4005 \\
-0.1279 & 0.9912 & 0.0352 & 17.8934 \\
0.1457 & -0.0163 & 0.9892 & -21.0986 \\
0 & 0 & 0 & 1
\end{bmatrix} \]

- for the bone \text{\textit{lhipjoint}}:

(a) 0 0 0 for the translation, and
(b) 0 0 0 for the rotation.

Using the same procedure, the transformation matrix \( T_{\text{\textit{lhipjoint}}} \) is therefore:

\[
T_{\text{\textit{lhipjoint}}} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

which is an identity matrix. The matrix \( T_{\text{\textit{lhipjoint}}} \) signifies that the bone \text{\textit{lhipjoint}} is an alias for the \text{\textit{root}} bone, and was created for clarity purposes in the skeleton hierarchy thus having no effect on the reconstruction process.

- for the bone \text{\textit{lfemur}}:

(a) 1.77779 -1.88989 0.48267 for the translation, and
(b) -7.1426 -13.4851 -25.1391 for the rotation.

Using the same procedure as with the \text{\textit{root}} bone (i.e. using \( R_{\text{\textit{lfemur}}} = R_{\gamma} \cdot R_{\beta} \cdot R_{\alpha} \) and creating the transformation matrix \( T_{\text{\textit{lfemur}}} \) by appending the \text{\textit{lfemur}} translation values), we arrive at:

\[
T_{\text{\textit{lfemur}}} = \begin{bmatrix}
0.9649 & 0.2109 & -0.1566 & 1.7778 \\
-0.1209 & 0.8859 & 0.4478 & -1.8899 \\
0.2332 & -0.4131 & 0.8803 & 0.4827 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

2. Multiply the transformation matrices to determine the global coordinates of each bone using the skeleton hierarchy.

- the \text{\textit{root}}:

Since the root bone is the first element of the skeleton hierarchy, the global position of the root bone is the position value itself:

\[
P_{\text{\textit{root}}} = L_{\text{\textit{root}}} = \begin{bmatrix}
4.4005 \\
17.8934 \\
-21.0986
\end{bmatrix}
\]
• the hipjoint:

\[ P_{\text{hipjoint}} = T_{\text{root}} \cdot T_{\text{hipjoint}} \]

\[
P_{\text{hipjoint}} = \begin{bmatrix}
  0.9810 & 0.1316 & -0.1424 & 4.4005 \\
  -0.1279 & 0.9912 & 0.0352 & 17.8934 \\
  0.1457 & -0.0163 & 0.9892 & -21.0986 \\
  0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
P_{\text{hipjoint}} = L_{\text{root}} = \begin{bmatrix}
  4.4005 \\
  17.8934 \\
  -21.0986
\end{bmatrix}
\]

Since the transformation matrix for the hipjoint bone is an identity matrix, the global coordinate for hipjoint is the same as root.

• the femur:

\[ P_{\text{femur}} = T_{\text{root}} \cdot T_{\text{hipjoint}} \cdot T_{\text{femur}} \]

\[
P_{\text{femur}} = P_{\text{hipjoint}} \cdot T_{\text{femur}}
\]

\[
P_{\text{femur}} = \begin{bmatrix}
  0.9810 & 0.1316 & -0.1424 & 4.4005 \\
  -0.1279 & 0.9912 & 0.0352 & 17.8934 \\
  0.1457 & -0.0163 & 0.9892 & -21.0986 \\
  0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
  0.9649 & 0.2109 & -0.1566 & 1.7778 \\
  -0.1209 & 0.8859 & 0.4478 & -1.8899 \\
  0.2332 & -0.4131 & 0.8803 & 0.4827 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  0.8975 & 0.3823 & -0.2201 & 5.8271 \\
  -0.2350 & 0.8366 & 0.4948 & 15.8099 \\
  0.3733 & -0.3924 & 0.8407 & -20.3313 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

\[ P_{\text{femur}} = L_{\text{femur}} = \begin{bmatrix}
  5.8271 \\
  15.8099 \\
  -20.3313
\end{bmatrix}
\]
Appendix E

MQF Audio Query Example

Instantiations

```xml
<mqf>
  <queryId>id_123</queryId>
  <from>client.uri</from>

  <query stream="true">
    <replyType>audio/mpeg</replyType>

    <item queryLevel="example">
      <mpeg7:MediaData64>AaBbCc/==</mpeg7:MediaData64>
    </item>
  </query>

</mqf>
```

Figure E.0.1: Updateable audio query step 1: The client sends an audio example using an attached binary sample encoded in Base64. The query level is set to query-by-example.
Figure E.0.2: Updateable audio query step 2: The server replies that there are 100 possible matches found that is similar to the sample given by the client.

```
<mqf>
  <queryId>id_123</queryId>
  <from>server.uri</from>
  <replies>
    <text>100 results matched</text>
  </replies>
</mqf>
```

Figure E.0.3: Updateable audio query step 2a: The result set that is kept in the server. This document is not sent to the client.

```
<mqf>
  <queryId>id_123</queryId>
  <from>server.uri</from>
  <replies>
    <reply>
      <item>
        <mpeg7:MediaUri>http://server/1.mp3</mpeg7:MediaUri>
      </item>
      <description>
        <mpeg7:Genre>Rock</mpeg7:Genre>
      </description>
      <index>1</index>
    </reply>
    <!-- 98 more reply items -->
    <reply>
      <item>
        <mpeg7:MediaUri>http://server/100.mp3</mpeg7:MediaUri>
      </item>
      <description>
        <mpeg7:Genre>Country</mpeg7:Genre>
      </description>
      <index>100</index>
    </reply>
  </replies>
</mqf>
```
APPENDIX E. MQF AUDIO QUERY EXAMPLE INSTANTIATIONS

Figure E.0.4: Updateable audio query step 3: The client’s FRU request to retrieve the genres in the result set shown in Figure E.0.3. Using FRU, the client does not need to receive the whole result set, however, the client can perform additional examination of the result set.

```xml
<FRU>
  <Src>id_123.mqf</Src>
  <Query>//mpeg7:Genre</Query>
</FRU>
```

Figure E.0.5: Updateable audio query step 4: The FUU replies from the FRU of Figure E.0.4. The server replied with the genre description of the result set that is kept in the server.

```xml
<FragmentUpdateUnit>
  <FUCommand>addNode</FUCommand>
  <FUContext>
    /mqf/replies/reply[0]/description
  </FUContext>
  <FUPayload>
    <mpeg7:Genre>Rock</mpeg7:Genre>
  </FUPayload>
</FragmentUpdateUnit>

<!-- 98 more fragment update units ... -->

<FragmentUpdateUnit>
  <FUCommand>addNode</FUCommand>
  <FUContext>
    /mqf/replies/reply[99]/description
  </FUContext>
  <FUPayload>
    <mpeg7:Genre>Country</mpeg7:Genre>
  </FUPayload>
</FragmentUpdateUnit>
```
Figure E.0.6: Updateable audio query step 5: The additional term to be added to the initial query in Figure E.0.1. The client requested that the server transmits items in the result set that belongs to the “Country” genre.

Figure E.0.7: Updateable audio query: The final query as seen by the server – the client requests that the previous result set (arrow 1) to be filtered to only include songs of the genre country (arrow 2).
Appendix F

Example of an HMML Authoring Tool

Figure F.0.1 depicted an example of an HMML authoring tool. Callouts in the Figure are:

1. The 3D representation of the motion being annotated.
2. Sagittal, coronal, and transverse movements plotted as described in chapter 5.
3. HMML element selection.
4. Motion playback control.