

Similarity, Retrieval, and Classification of Motion Capture Data

Dissertation

zur

Erlangung des Doktorgrads (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Tido Röder

aus

Osnabrück

Bonn, 7. September 2006

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn

http://hss.ulb.uni-bonn.de/diss_online

elektronisch publiziert.

1. Referent: Prof. Dr. Michael Clausen
2. Referent: Prof. Dr. Daniel Cremers
Tag der Promotion: 2. März 2007
Erscheinungsjahr: 2007

Similarity, Retrieval, and Classification of Motion Capture Data

Tido Röder

Abstract

Three-dimensional motion capture data is a digital representation of the complex spatio-temporal structure of human motion. Mocap data is widely used for the synthesis of realistic computer-generated characters in data-driven computer animation and also plays an important role in motion analysis tasks such as activity recognition. Both for efficiency and cost reasons, methods for the reuse of large collections of motion clips are gaining in importance in the field of computer animation. Here, an active field of research is the application of morphing and blending techniques for the creation of new, realistic motions from prerecorded motion clips. This requires the identification and extraction of logically related motions scattered within some data set. Such content-based retrieval of motion capture data, which is a central topic of this thesis, constitutes a difficult problem due to possible spatio-temporal deformations between logically related motions. Recent approaches to motion retrieval apply techniques such as dynamic time warping, which, however, are not applicable to large data sets due to their quadratic space and time complexity. In our approach, we introduce various kinds of *relational features* describing boolean geometric relations between specified body points and show how these features induce a temporal segmentation of motion capture data streams. By incorporating spatio-temporal invariance into the relational features and induced segments, we are able to adopt indexing methods allowing for flexible and efficient content-based retrieval in large motion capture databases.

As a further application of relational motion features, a new method for fully automatic motion classification and retrieval is presented. We introduce the concept of *motion templates* (MTs), by which the spatio-temporal characteristics of an entire motion class can be learned from training data, yielding an explicit, compact matrix representation. The resulting class MT has a direct, semantic interpretation, and it can be manually edited, mixed, combined with other MTs, extended, and restricted. Furthermore, a class MT exhibits the characteristic as well as the variational aspects of the underlying motion class at a semantically high level. Classification is then performed by comparing a set of precomputed class MTs with unknown motion data and labeling matching portions with the respective motion class label. Here, the crucial point is that the variational (hence uncharacteristic) motion aspects encoded in the class MT are automatically masked out in the comparison, which can be thought of as locally adaptive feature selection.

Keywords: motion capture, similarity, relational feature, content-based, retrieval, classification, indexing, motion template, segmentation, computer animation, multimedia information retrieval

Danksagung / Acknowledgements

Viele Menschen haben zum Gelingen dieser Arbeit beigetragen, sei es durch hilfreiche Diskussionen, wertvolle Hinweise, Unterstützung jeglicher Art, Großzügigkeit, persönlichen Einsatz, Freundschaft, harte Arbeit oder entgegengebrachtes Vertrauen. Allen voran danke ich meiner lieben Frau *Loziana*, die mir in schwierigen Zeiten stets eine Motivation war, für ihre Zuneigung, ihre Unterstützung und ihre Geduld. Ebenso danke ich meinem Vater, *Heiner Röder*, meinen Geschwistern *Gunther, Inga, Almut, Sita* samt Anhängen, meinen Schwiegereltern, *Diana* und *Lulzim Cara* und meinem Schwager, *Erlind Cara*. Besonderer Dank gilt auch meinem Onkel, *Lothar Uhlenbusch*, für die Vorfinanzierung des Drucks.

Bei der gesamten Arbeitsgruppe von Prof. Michael Clausen bedanke ich mich für die angenehme, produktive Zeit, die immerhin von meiner Anstellung als studentische Hilfskraft im Jahr 2002 bis zum Jahr 2006 angedauert hat. *Michael Clausen* danke ich neben der ausgezeichneten Betreuung, der allzeitigen Ansprechbarkeit und der guten Zusammenarbeit für die Bereitstellung der äußeren Bedingungen in Form eines vorzüglichen Arbeitsplatzes, immer wieder neu erkämpften SHK-, WHK- und WiMi-Stellen und der Ermöglichung von Konferenzbesuchen. Durch die intensive und motivierende Zusammenarbeit mit *Meinard Müller* habe ich nicht nur dazugelernt, sondern auch viel Spaß gehabt und einen guten Freund gewonnen. Danke! Auch meinen Kollegen und Korrekturlesern *Frank Kurth, Rolf Bardeli, Andreas Ribbrock, Frank Schmidt, Christian Fremerey, David Damm, Daniel Wolff, Bastian Demuth* und *Hendrik Ewe* danke ich für ihre Freundschaft, die angenehme Zusammenarbeit, ihre Unterstützung, für Diskussionen, Bashos, Kekse, Mohnkuchen und viele Kleinigkeiten, die den Arbeitsalltag lebenswert machen.

Auf kompetente und liebenswerte Weise haben *Martina Doelp* und *Heidi Georges-Hecking* den Umgang mit Verwaltungsdingen zu einem großen Teil von mir fernhalten können. Herzlichen Dank! Ebenso war mir die Systemgruppe Altbau eine große Hilfe: der Support durch *Peter Lachart* und *Thomas Fuchs* hat mir nicht nur nach dem Altbaubrand viel Zeit erspart.

Prof. Andreas Weber gilt besonderer Dank für die Inspiration zu dieser Arbeit und seiner positiven Einschätzung bezüglich der Relevanz für die Computeranimation. Erst durch seinen Anstoß wurden wir auf das Thema Bewegungsdaten aufmerksam. Herzlichen Dank auch an *Björn Krüger* und *Daniel Goldbach* für die Zusammenarbeit bei der Aufnahme und Aufbereitung der Bewegungsdaten, sowie an *Arno Zinke* für Diskussionen.

Prof. Bernd Eberhardt sowie *Harun Celebi* und *Jochen Bomm* von der Hochschule der Medien, Stuttgart, danke ich für die Bereitstellung der Kapazitäten zur Aufnahme und Aufbereitung der Bewegungsdaten und für die gute Zusammenarbeit.

Prof. Daniel Cremers danke ich für seine Bereitschaft zur Erstellung des Zweitgutachtens, ebenso bei *Prof. Gerald Sommer* von der Universität Kiel für die Erstellung eines Gutachtens für mein Promotionstipendium. *Bodo Rosenhahn* danke ich für hilfreiche Diskussionen, Hinweise und Literaturquellen.

Der Studienstiftung des Deutschen Volkes gilt mein Dank für die finanzielle Unterstützung. Ebenso bedanke ich mich bei *Prof. Heinz-Josef Fabry* für die regelmäßigen Treffen im Casa del Gatto und für die interessanten Ausflüge.

I would like to thank *Professor Jessica Hodgins* and the Carnegie Mellon mocap lab for providing the skeleton files required for the C3D to ASF/AMC conversion. Some of the data used in this project was obtained from `mocap.cs.cmu.edu`, which was created with funding from NSF EIA-0196217.

Contents

1	Introduction	1
1.1	Overview and Contribution	2
1.2	Motion Capture Data	5
1.3	Kinematic Chains	8
2	Similarity of Motions	19
2.1	Human Perception of Motions	20
2.2	Global Aspects of Similarity	21
2.3	Content vs. Style	22
2.4	Logical vs. Numerical Similarity	24
2.5	Local Similarity Measures	28
2.6	A Semi-Local Similarity Measure	33
2.7	Global Similarity Measures	34
2.8	Examples and Comparative Evaluation	35
3	Relational Motion Features	41
3.1	Examples of Relational Features	41
3.2	Temporal Segmentation	46
3.3	Feature Design	49
3.4	Experimental Feature Sets	66
3.5	Measuring Motion Similarity with Relational Features	68
4	Efficient Index-Based Motion Retrieval	71
4.1	Query and Hit Concepts	71
4.2	Efficient Computation of Hits	75
4.3	A Retrieval System Using the Query-By-Example Paradigm	79
4.4	Experimental Results	84
4.5	Discussion and Possible Extensions	86
4.6	Related Work	88
5	Motion Templates for Retrieval and Classification	93
5.1	Learning MTs from Example Motions	93
5.2	MT-based Matching for Retrieval and Classification	99
5.3	Related Work	108
6	Conclusions and Future Work	111
6.1	Future work	112

A	Representing Rotations of \mathbb{R}^3	113
A.1	Rotation Matrices	114
A.2	Axis/Angle	116
A.3	Quaternions	117
A.4	Exponential Map	122
A.5	Euler Angles	123
A.6	Compendium of Conversion Formulas	137
B	Motion Smoothing with Quaternion Filters	145
B.1	Weighted Averages in \mathbb{R}^4	146
B.2	Spherical Weighted Averages	146
B.3	Orientation Filters	148
B.4	Evaluation of Smoothing Strategies	152
C	2D Alignment of 3D Point Clouds	155
D	Motion Capture File Formats	159
D.1	The ASF/AMC Format	159
D.2	The BVH Format	164
D.3	Mapping Mocap Data to the Standard Skeleton	167
D.4	The C3D Format	167
	List of Figures	169
	List of Tables	173
	Bibliography	175
	Index	185

Chapter 1

Introduction

In our everyday lives, we are constantly confronted with human motion: we watch other people as they move, and we perceive our own movements. Motion, in terms of physics, is a change in the position of a body with respect to time. Since the human body is not simply a rigid block but rather a complex aggregation of flexibly connected limbs and body parts, human motion can have a very complex spatio-temporal structure. Our brain is highly specialized in analyzing and understanding such complex motion information in real time, using as input its multi-modal sensor arrays: the eyes, the ears, and, in the case of self-perception, the somatosensory haptic and position senses [Gol02].

Let us move from the real world to the digital domain by considering modern *motion capture* technology, which is capable of accurately digitizing a motion's spatio-temporal structure for further processing on a computer. On the one hand, the resulting *mocap* data can be used for motion *synthesis* in data-driven computer animation. On the other hand, it provides the basis for motion *analysis* in applications such as motion recognition, retrieval, and classification. Regarding both synthesis and analysis, the human perceptual system sets high quality standards: our vast experience with natural human motion enables us to easily detect unnatural or synthetic motions, and even low-level motion analysis tasks that seem so easy to us—think of segmenting continuous motion into basic behavioral units—prove to be a challenging task to a computer.

In the field of computer animation, the dominating technique for 3D character animation has long been the traditional *keyframing* method, which is also employed in the generation of 2D cartoons. Keyframing is the process of sketching out a motion by specifying certain key poses and then interpolating the missing poses to obtain a continuous motion. For keyframed animations to look realistic, the animator requires a lot of time and experience. Hence, keyframing is very expensive, and the results are not always perfectly convincing. As an advantage of keyframing, the animator has full control over the character. The modern alternative to keyframing is the use of mocap data to drive the virtual character, resulting in highly realistic animations. A well-known example is the *Gollum* character from the 2002 motion picture *The Lord of the Rings: The Two Towers*, which has been animated using mocap data recorded from the actor Andy Serkis. The combination of the realistic looks of the character itself and the mocap-generated movements creates the almost perfect illusion of a sinister monster.

Two disadvantages are associated with mocap data. First, the recording equipment is very expensive, and second, mocap data is difficult to edit without causing visible artifacts.

As a consequence, captured sequences that do not exactly match the director’s intentions often have to be recaptured. Therefore, much attention in the computer graphics research community has been directed at developing techniques for *motion reuse* via blending and morphing. Starting with basic motion editing techniques [BW95, WP95], many different methods have been suggested to create new, realistic motions from prerecorded motions, see, for example, [GP00, PB02, AFO03, KG03, KG04, CH05, ZMCF05] and the references therein. Techniques for motion reuse are often based on large motion databases, requiring efficient retrieval, browsing, and annotation methods in order to fully exploit the variety of available motion clips.

This leads us to the field of *motion analysis*, which constitutes a central topic of this thesis. Only recently, motion capture data has become publicly available on a larger scale (e. g., the CMU database [CMU03]), reinforcing the demand for efficient indexing and retrieval methods. Our colleagues from Stuttgart Media University (HdM), who frequently produce computer-generated movies involving mocap data, report on the typical “data graveyard” problem: they have vast collections of mocap data without a way of knowing what kinds of motions they have and where and how to search for certain motions. Annotating mocap files by hand using descriptive filenames is tedious and only provides a partial solution to this problem.

Here, *content-based retrieval* of motion capture data comes into play. The term refers to techniques that can find motion clips by using only the data itself, without resorting to manually generated metadata. However, most current techniques for content-based retrieval of mocap data pay the price for the inherent complexity of human motion in terms of computational complexity. The underlying question is the difficult problem of *how to measure similarity* between motions or how to *compare* motions in a meaningful way. Since motions that are perceived by a human as *logically related* may differ by significant spatio-temporal deformations, some means for absorbing such deformations have to be built into the matching technique, which causes high computational overhead. As a major contribution of the present thesis, we introduce *relational features* in conjunction with *temporal segmentation*. By virtue of these concepts, we can absorb spatio-temporal deformations at the feature level while using efficient, index-based retrieval methods that are similar to full-text retrieval. The resulting content-based retrieval system is another contribution of the present work.

As a further application of relational features, we propose a new concept for capturing the spatio-temporal characteristics of an entire motion class in one explicit, compact matrix representation called *motion template* (MT). An MT-based matching technique then provides the basis for flexible and fully automatic retrieval and classification of mocap data, with possible use in tasks such as activity recognition.

1.1 Overview and Contribution

This section has been written to give the “big picture” of what our contributions are and provides the reader with a guide to proceeding through this thesis. Throughout the thesis, we use the abbreviations Chap. (Chapter), App. (Appendix), Sect. (Section), Fig. (Figure), Tab. (Table), Eqn. (Equation). Alternatively, equations are often referenced by an equation number in parentheses such as (1.1). The symbol \square denotes the end of a proof, whereas \diamond denotes the end of a definition.

Motion Capture Data. The remainder of Chap. 1 is devoted to motion capture data and its underlying structure. After giving a formal definition, we briefly touch in Sect. 1.2 on the history of mocap data and on modern recording equipment. We then explain how mocap data is recorded in actual studio sessions. In Sect. 1.3, we introduce kinematic chains, which are commonly used to model the human skeleton.

Similarity of Motions. Chap. 2 deals with various aspects of similarity that make the comparison of motions a difficult task. We start in Sect. 2.1 by a short overview of the research in the field of *perception*, an important area of psychology. In Sect. 2.2–2.4, we then discuss a large number of example motions, shedding light on global aspects of similarity, on the issue of separating content from style, and on the difficult relation between *logical* and *numerical* similarity. In Sect. 2.5, we exemplarily introduce two *local* similarity measures. After discussing a *semi-local* similarity measure in Sect. 2.6, the local similarity measures of Sect. 2.5 are put to use in *global* similarity measures based on the computationally expensive technique of *dynamic time warping* (DTW). We conclude this chapter with examples and a comparative evaluation in Sect. 2.8.

Relational Motion Features. Having raised the reader’s awareness of the problems that may occur when comparing motions by means of numerical features such as 3D joint trajectories, Chap. 3 then introduces our new concept of relational features and feature-induced temporal segmentations. These features and the induced segments exhibit several properties that make them well-suited for efficient and fault-tolerant comparison of motions:

1. Relational feature functions are largely invariant to various kinds of spatial transformations and deformations of poses.
2. Induced feature sequences are largely invariant to various kinds of spatio-temporal transformations and deformations of motions.
3. Sequences of boolean feature vectors can be viewed as strings over the finite alphabet $\Sigma := \{0, 1\}^f$.

The key property of relational feature sequences is that they already absorb spatio-temporal deformations at the feature level, forming a largely invariant “fingerprint” of a motion. This enables the use of discrete matching strategies, which are particularly efficient.

Chap. 3 pursues an introduction-by-example approach by discussing a number of examples of relational motion features in Sect. 3.1. Then, we explain in Sect. 3.2 how a mocap data stream can be divided into temporal segments derived from the corresponding feature sequence. The following Sections 3.3 and 3.4 deal with the design of relational features and explain the features that have been used in our experiments. Finally, we compare in Sect. 3.5 a global similarity measure based on relational features to the numerically-based similarity measures of Sect. 2.8

Efficient Index-Based Motion Retrieval. In Chap. 4, we explain how our relational features can facilitate efficient content-based retrieval in large mocap databases. Various query and hit concepts are introduced in Sect. 4.1. We then explain how such hits can be efficiently computed using an index structure based on inverted lists, see Sect. 4.2. The following section, Sect. 4.3, describes a concrete retrieval system based on the query-by-example paradigm that

uses the query and hit concepts introduced in the preceding sections. Experimental results are provided in Sect. 4.4. The chapter is closed by a discussion in Sect. 4.5 and a summary of related work in Sect. 4.6.

At this point, we would like to encourage the reader to watch our video *Efficient Content-based Retrieval of Motion Capture Data*, which illustrates the concepts introduced in our publications [MRC05b] and [MRC05c], see also Chap. 3 and Chap. 4. The video can be found on our web site [MRC05a] and on the CD-ROM accompanying this thesis. Additional material pertaining to our publication [DRME06a] is available at [DRME06b].

Motion Templates for Retrieval and Classification. It turns out that index-based motion retrieval is capable of efficiently delivering high-quality retrieval results, but under one condition: the user has to incorporate some prior knowledge about the intended motion hits by selecting suitable features from a given set of relational features. This manual interaction is a disadvantage in view of fully automatic retrieval, as required for automatic motion reuse applications.

In Chap. 5 we introduce the new concept of *motion templates* (MT), which capture the the spatio-temporal characteristics of an entire motion class in an explicit, compact matrix representation. In addition this matrix representation has a direct, semantic interpretation. An MT can be learned from training data, edited, modified, mixed, combined with other MTs, extended, and restricted, thus providing a great deal of flexibility. The important idea in MT-based motion matching is to fully automatically exclude the variational aspects of a motion class in the comparison—allowing us to identify logically related motions even in the presence of large variations. By “variational aspects” of a motion class, we mean certain components that may change significantly between different realizations of a motion.

Sect. 5.1 describes how to learn MTs from suitable training data pertaining to a certain motion class. Then, Sect. 5.2 describes in detail the above mentioned matching technique, which masks out the variational aspects of the respective motion class. This technique is then employed for DTW-based retrieval and classification, where we introduce an efficient, index-based preprocessing step using *keyframes* to speed up the DTW computation. Often, the words “recognition”, “understanding”, and “annotation” are used in a similar sense as “classification”. In the following, we try to avoid the former two expressions, since they imply that the goal is to acquire deeper knowledge about the purpose of a motion. To us, motion classification and annotation refer to the process of assigning class labels to certain frames of an unknown motion. The chapter closes with a discussion of related work in Sect. 5.3.

We recommend to watch our video *Motion Templates for Automatic Classification and Retrieval of Motion Capture Data*, which accompanies our publication [MR06b], see also Chap. 5. It can be found at [MR06a] and on the CD-ROM enclosed with this thesis.

Appendices. The four appendices contain supplemental technical details. In App. A, we give a detailed treatment of various representations for rotations in \mathbb{R}^3 , including Euler angles, quaternions, and the quaternionic exponential. Rotations play an important role in skeleton-based mocap data, and parsing and manipulating mocap files requires some knowledge of rotational representations. App. B reviews and compares several techniques for motion smoothing based on quaternion filters. In App. C, we deal with the alignment of 3D point clouds, a sub-task arising in one of the local similarity measures discussed in Chap. 2. Finally, we describe in App. D the details of two common mocap file formats, ASF/AMC and BVH.

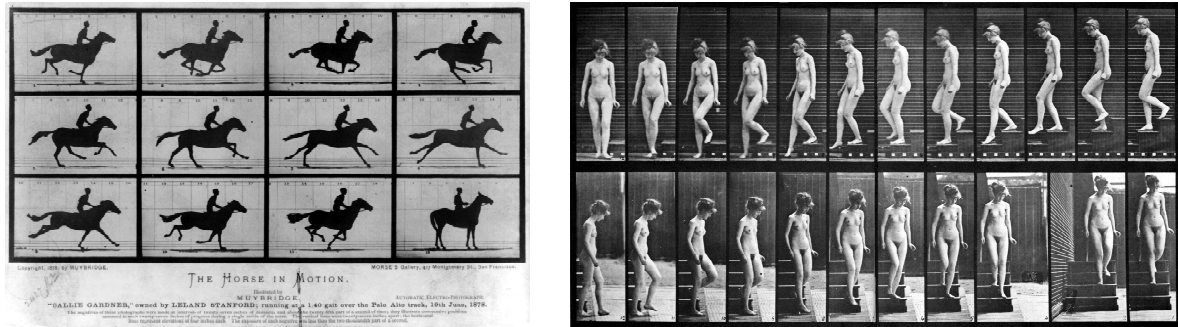


Figure 1.1. Eadward Muybridge’s electro-photography. **Left:** *The horse in motion*, taken from [Muy87], 1887. **Right:** *Woman walking downstairs* (right to left), taken from [Muy01], 1901.

1.2 Motion Capture Data

Using motion capture techniques, one can record and store a digital approximation of a human actor’s three-dimensional full-body movements. Besides the three spatial dimensions, motions also depend on time. Since only a limited number of 3D body points can be captured in their temporal evolution, and since the motion of neighboring points on the same limb are highly correlated, one often restricts the set of captured points to certain joints of the human skeleton. This leads to the following definition.

Definition 1.1 (Motion capture data stream)

A mocap data stream D with joint set J is a time-dependent sequence of frames or poses,

$$D : [1 : T] \rightarrow \mathcal{P} \subset \mathbb{R}^{3 \times |J|}, \quad (1.1)$$

where each pose $P \in \mathbb{R}^{3 \times |J|}$ specifies the 3D coordinates of the joints at a certain point in time. The j^{th} column of P , denoted as P^j , corresponds to the 3D coordinates of joint $j \in J$. The set $[1 : T]$ represents the time axis (for a fixed sampling rate), and \mathcal{P} denotes the set of poses, also referred to as the pose space. The curve described by the 3D coordinates of a single joint $j \in J$ is termed trajectory and denoted as $D^j : [1 : T] \rightarrow \mathbb{R}^3$. \diamond

The joint set, J , often corresponds to the nodes of a so-called *kinematic chain* model, which may be thought of as a simplified version of the human skeleton, see Sect. 1.3 for more details. Note that we only focus on full-body motion capture, excluding details such as facial expressions.

1.2.1 Motion Capture Technology

Analog precursors to modern digital motion capture equipment have been developed since the late 19th century, mainly for the purpose of kinematic and biomechanical motion analysis, see [MCA06]. In 1872, one of the fathers of *chronophotography*, Eadward Muybridge, developed a photographic array consisting of up to 36 cameras, with which he could capture short 2D image sequences at frame rates of about 25 Hz. He named his technique *electro-photography* because the cameras were triggered electrically. Fig. 1.1 shows two of his famous image sequences. The sequence *Horse in motion* was commissioned by Leland Stanford, the founder of Stanford University, in the context of a bet: the objective was to prove that there are periods during a

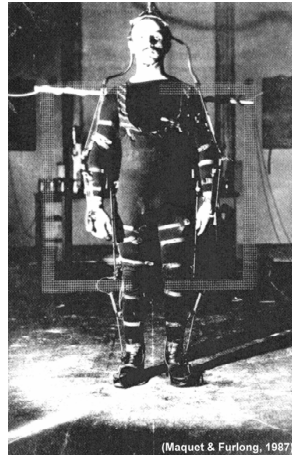


Figure 1.2. Braune and Fischer’s motion capture apparatus, around 1892. Taken from [BF87].

horse’s galloping cycle when all four legs lift off the ground simultaneously. Fig. 1.1 shows that Stanford won his bet.

Inspired by Muybridge’s work, Braune and Fischer [BF91, BF94, Fis99] invented the first 3D optical motion capture apparatus, which used active markers (emitting electric sparks) attached to the actor’s body in combination with an electro-photographic stereo setup, see Fig. 1.2. On the resulting photographic plates, they were able to measure the 2D trajectories at a spatial resolution of $1\ \mu\text{m}$, allowing for high-precision 3D reconstructions. All required computations had to be performed by hand, hence the evaluation of a single captured sequence would take several months.

In the early 20th century, after cinematography as we know it today had been developed, *rotoscopy* was invented as a technique for the production of animated figures. Rotoscopy is the process of tracing photographic sequences (by an artist or by a computer) to produce cartoon-like figures that move like real characters. Thus, rotoscopy can be seen as a simple precursor to computer-generated animations using 3D mocap data.

With the dawn of the modern computer age, the first digital mocap systems were developed in the early 1980s, see [Stu94]. The driving force behind this development was once again biomechanical motion analysis. Mocap techniques for computer animation quickly followed, eventually leading to the modern, commercial systems that are now in wide use. In the following, we give a short overview of modern 3D mocap equipment. For a comparison of the accuracy of different mocap systems, we refer to [Ric99].

Optical Mocap. The most common mocap systems are *optical*, marker-based systems. These systems use a set of active (light-emitting) or passive (retro-reflective) markers that are attached to the actor’s body. An array of 6–12 or even more digital cameras tracks the markers at high spatial and temporal resolution, yielding 2D marker data from which the 3D marker trajectories can be computed in real time, see Fig. 1.3. The resulting data has submillimeter precision at frame rates of 60–240 Hz. A typical example of an optical mocap system is the Vicon system [Vic06]. While being very expensive, optical mocap systems have the benefit of being precise and scalable (more cameras can easily be added to enhance precision and enlarge the capture volume). The disadvantage of optical mocap systems is that they will typically

not be operable in outdoor settings since the cameras and tracking algorithms cannot handle bright sunlight.

Magnetic Mocap. Magnetic mocap systems such as Ascension [Asc06] are also based on body-worn markers, but the measuring method uses a magnetic field. The position and orientation of the markers within the magnetic field can be determined in real time. Magnetic mocap is less expensive and can operate outdoors, but it is very sensitive to distortions of the magnetic field as caused by ferrous materials or electric appliances.

Mechanical Mocap. Mechanical mocap systems use body-attached sensors that directly measure the movement of the joints. For example, the Gypsy system [Gyp06] uses an exoskeleton to drive potentiometers providing real-time angle data at very high frame rates. Similarly, the ShapeWrap [Sha06] technique is based on joint-attached, flexible extension meters. The advantage of mechanical systems is their low price, their high reliability, and the fact that they can be applied in outdoor settings. On the other hand, mechanical systems are often based on a rather crude model of the human skeleton, yielding less accurate data. Furthermore, the exoskeleton may interfere with the actor's motions.

1.2.2 Capturing Motions

Motion capture is a complex process that consists of several stages and requires careful preparation. To give an impression of the typical workflow and the potential sources of errors in mocap data, we outline the steps that were taken to record the main corpus of test data used in this thesis, see also Sect. 5.1.2.

The first task was to create a detailed script describing the roughly 50 different motions that were to be recorded by five different actors. These motions were rehearsed several times before the recording sessions. Typically, several motions were grouped into a *take*, where each take had a duration of at most one minute.

For the actual recording, we used a twelve-camera passive optical mocap system by Vicon [Vic06] comprising six infrared and six visible red cameras, see Fig. 1.3. Setting up this system from scratch requires roughly two hours of an expert's time to facilitate proper camera placement, wiring, software configuration, and calibration of the capture volume for metric 3D reconstruction of marker positions at millimeter precision. Since the accuracy of measurements is highly sensitive to the absolute and relative orientation of the cameras, the capture volume must be recalibrated on a regular basis.

Once the system is set up, each actor is equipped with a contrast-enhancing black nylon suit to which passive markers are attached in a specific pattern using velcro pads. Additional markers are worn on head- and wristbands and on special gloves, for a total of 44 markers. Exactly placing the markers over bony landmarks so as to minimize skin shift and other undesirable effects is a difficult task and requires expert knowledge. After marker placement, a range-of-motion calibration has to be performed for each actor. Here, a prescribed sequence of joint movements to their maximum extents is captured. This information then aids the real-time process of assigning and tracking the identities of the indistinguishable passive markers. Also, a static T-pose (Fig. 1.3) is captured, from which the semantics of each marker is initialized by manual assignment.

During the recording sessions, it turned out that the velcro attachment of the markers was not very stable, leading to frequent loss of one or more markers due to rapid motions



Figure 1.3. Passive optical motion capture system based on retro-reflective markers attached to the actor’s body. The markers are tracked by an array of six to twelve calibrated high-resolution cameras arranged in a circle around the capture volume, which has a radius of approx. 2.5 m in this example.

or ground contact, which usually necessitated a new take for the respective motion. The net total recording time for five actors was about 8 hours, where each actor repeated each take several times.

The resulting raw 3D mocap data had to be cleaned by an expert, which took more than a week. Cleaning a mocap data stream amounts to a manual inspection for lost markers, markers that got mixed up by the tracking algorithm, and occluded markers. Each of these situations has to be resolved by hand, usually guided by semi-automatic detection, interpolation, and gap filling methods. Finally, after the cleaning stage, the data was fitted to a skeletal model using commercial software. Besides taking considerable time, this step proved to be highly heuristic and extremely hard to control in all details. As a result, typical skeletal mocap data is by far not an exact representation of the actor’s motion even though it might be visually pleasing. This condition is widely accepted in computer animation, but of course intolerable for biomechanical or medical applications. The topic of skeletal fitting is still a field of active research, see [dATS06, OBBH00].

1.3 Kinematic Chains

In computer animation, kinematic chains are a widely used tool for modeling complex, three-dimensional movable objects such as the human skeleton, see Fig. 1.4 for an example. Basically, a kinematic chain is a hierarchical system of rigid bodies (here: line segments representing the bones) that are connected by movable joints with various degrees of freedom.

We focus on *open* kinematic chains, in which no cycles occur in the connection hierarchy—that is, we require the connection hierarchy to be tree-structured. Starting from a *root* object, child objects are attached, which may in turn have further child objects, and so on. If a parent object moves by means of its degrees of freedom, the entire subtree below the parent object, treated as a single rigid object, moves along. In other words, transformations of a child object are meant to take place *relative* to a coordinate system that is fixed at the parent object. We

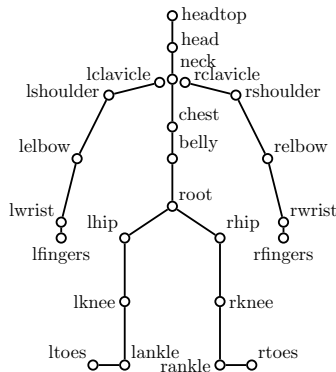


Figure 1.4. Skeletal kinematic chain model consisting of rigid *bones* that are flexibly connected by *joints*, which are highlighted by circular markers and labeled with joint names.

discuss some details on coordinate systems in Sect. 1.3.1 before giving a formal definition of kinematic chains in Sect. 1.3.2.

If we know the state of all joints in a kinematic chain, we can compute the 3D position and rotation of all involved rigid bodies relative to the root, i. e., up to an absolute position and rotation. This computation is known as *forward kinematics* and will be described in Sect. 1.3.2. As a consequence, the collective information about the state of all joints in a kinematic chain—also referred to as the kinematic chain’s *parameters* or *degrees of freedom (DOF)*—is a representation of the rigid body system that is essentially invariant to the group of similarity transformations in 3D. This invariance property is often desirable for motion synthesis and analysis tasks.

The joints of a kinematic chain are either *revolute joints* with up to three rotational DOF, or they are *prismatic joints* with up to three translational DOF, or they are a combination of both, totaling up to six DOF. The extreme case of a six DOF joint can be thought of as a fully rotatable platform with a positioning device such as a gantry attached to it.

Most joints of the human skeleton can be reasonably well approximated by purely revolute joints, so we will not further consider prismatic joints, see Sect. 1.3.4 for more details on this assumption. The state of a revolute joint is described by some form of rotational representation such as a set of Euler angles, a unit quaternion, or a rotation matrix, cf. App. A. For algebraic simplicity, we choose to describe rotations in terms of rotation matrices at this point. We derive *animated* kinematic chains as sequences of kinematic chains with time-dependent parameters in Sect. 1.3.3.

1.3.1 Local Coordinate Systems

Since a kinematic chain defines a hierarchical composition of rotations and translations, it is helpful to think of the intermediary results at each level of the hierarchy in terms of local (or *affine*) coordinate systems. A *local coordinate system* in \mathbb{R}^3 , $\mathcal{L} = (o; x_1, x_2, x_3)$, is determined by an origin $o \in \mathbb{R}^3$ and three vectors $x_1, x_2, x_3 \in \mathbb{R}^3$, such that $(x_i - o)_{i=1}^3$ forms a basis of \mathbb{R}^3 . Any vector $v \in \mathbb{R}^3$ can then be expressed as a linear combination

$$v = o + \sum_{i=1}^3 \alpha_i (x_i - o), \quad (1.2)$$

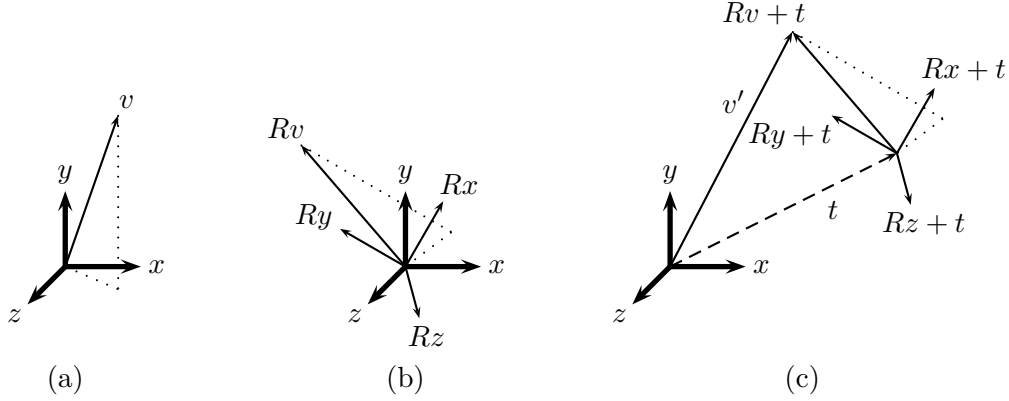


Figure 1.5. Action of the Euclidean motion $T_{(R,t)}$ of (1.3) on a vector v and on a reference frame that is rigidly attached to v . The vector v' is the image of v under $T_{(R,t)}$.

where $\overset{\mathcal{L}}{v} := (\alpha_1, \alpha_2, \alpha_3)^\top \in \mathbb{R}^3$ are the *local coordinates* of v with respect to \mathcal{L} . In the following, we will only consider the special case that $(x_i - o)_{i=1}^3$ forms a right-handed, orthonormal basis.

We fix a *world coordinate system* $\mathcal{W} := (o; e_1, e_2, e_3)$, where the e_i are the vectors of the standard basis of \mathbb{R}^3 . The key to understanding the semantics of a kinematic chain are coordinate transforms from local coordinate systems \mathcal{L} to the world coordinate system \mathcal{W} .

In our scenario of kinematic chains, local coordinate systems do not explicitly arise in the form $\mathcal{L} = (o; x_1, x_2, x_3)$. Instead, we are given a pair $(R, t) \in \text{SO}_3 \times \mathbb{R}^3$ describing a coordinate transform from the local coordinate system to the world system by means of the Euclidean motion

$$\begin{aligned} T_{(R,t)} : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ v &\mapsto Rv + t, \end{aligned} \quad (1.3)$$

where $R \in \text{SO}_3$ is a rotation matrix and $t \in \mathbb{R}^3$ is a translation vector. The explicit form of the coordinate system $\mathcal{L} = (o; x_1, x_2, x_3)$ can be read off from the pair (R, t) in the following easy way: the origin o equals t , and the i^{th} column of the matrix $R = (r_1 \ r_2 \ r_3)$ equals $(x_i - o)$.

Fig. 1.5 illustrates the action of a Euclidean motion: we start with a vector $v \in \mathbb{R}^3$ expressed in some coordinate system, say $(0; x, y, z)$, see Fig. 1.5 (a). $T_{(R,t)}$ first rotates v by R (as shown in Fig. 1.5 (b) together with the rotated coordinate frame $(0; Rx, Ry, Rz)$, which is thought to be moving along with v) and then translates it by t (Fig. 1.5 (c)), yielding the result vector $v' \in \mathbb{R}^3$. In plain words, this sequence of transformations can be interpreted as embedding the local coordinate system within the world system.

Now, suppose we are given a coordinate vector $\overset{\mathcal{L}}{v}$ expressed in the local coordinate system \mathcal{L} , which is defined relative to the world system by the pair $(R_{\mathcal{L}}^{\mathcal{W}}, \overset{\mathcal{W}}{t}) \in \text{SO}_3 \times \mathbb{R}^3$. That is, $\overset{\mathcal{W}}{t}$ is the origin of the local coordinate system expressed in world coordinates, and $R_{\mathcal{L}}^{\mathcal{W}}$ describes the rotation of the local coordinate axes relative to the world coordinate axes. The world coordinates for $\overset{\mathcal{L}}{v}$ are then simply

$$\overset{\mathcal{W}}{v} = R_{\mathcal{L}}^{\mathcal{W}} \overset{\mathcal{L}}{v} + \overset{\mathcal{W}}{t}. \quad (1.4)$$

It is always possible to add another level in a hierarchy of coordinate systems, replacing the world system, \mathcal{W} , by a new world system, \mathcal{W}' , see also Fig. 1.6. Suppose \mathcal{W} is defined relative to the new world system \mathcal{W}' by means of $(R_{\mathcal{W}}^{\mathcal{W}'}, \overset{\mathcal{W}'}{t}) \in \text{SO}_3 \times \mathbb{R}^3$. Applying Eqn. (1.4), the new

world coordinates of v are

$${}^{w'}v = R_{\mathcal{W}}^{w'} {}^wv + {}^w t = R_{\mathcal{L}}^{w'} (R_{\mathcal{L}}^w {}^{\mathcal{L}}v + {}^w t) + {}^w t = R_{\mathcal{W}}^{w'} R_{\mathcal{L}}^w {}^{\mathcal{L}}v + R_{\mathcal{W}}^{w'} {}^w t + {}^w t. \quad (1.5)$$

Observe that the rotations are successively applied in the sequence as they appear when moving from the local coordinate system towards the world coordinate system.

1.3.2 Kinematic Chains and Forward Kinematics

The following definition provides a formal framework for the ideas that have been discussed in the introduction to this chapter.

Definition 1.2 (Kinematic chain)

An (open) kinematic chain $C = (J, r, B; R_r, t_r, (R_b)_{b \in B}, (t_b)_{b \in B})$ is a rooted, edge-attributed tree where

- J is a set of vertices (the joints),
- $r \in J$ is the root,
- $B \subset J \times J$ is a set of edges that are directed away from the root (the bones),
- $R_r \in \text{SO}_3$ is the root rotation,
- $t_r \in \mathbb{R}^3$ is the root translation,
- $(R_b)_{b \in B} \in (\text{SO}_3)^{|B|}$ defines a relative joint rotation for each bone $b \in B$, and
- $(t_b)_{b \in B} \in (\mathbb{R}^3)^{|B|}$ defines a relative joint translation for each bone $b \in B$.

For a bone $(j_1, j_2) \in B$ we say that j_1 is the proximal joint (closer to the root) and j_2 is the distal joint (further away from the root). The parent of a joint $j \in J \setminus \{r\}$ is denoted by $p(j)$. The leaves of a kinematic chain are also referred to as end effectors. \square

Interpretation of Kinematic Chains. A kinematic chain $C = (J, r, B; R_r, t_r, (R_b)_{b \in B}, (t_b)_{b \in B})$ provides hierarchical instructions about assembling and positioning the bones and joints in 3D, a process known as *forward kinematics*. Forward kinematics successively assigns local coordinate systems to the distal joints of all bones by means of the relative joint rotations, $(R_b)_{b \in B}$, and the relative joint translations, $(t_b)_{b \in B}$. Each coordinate system at a distal joint is specified relative to the coordinate system at the respective proximal joint, which is assumed to have been computed in a previous step. Thus, a bone $b \in B$ effectively rotates at its proximal joint by $R_b \in \text{SO}_3$, and the distal joint follows this rotation together with the entire subtree rooted at the distal joint. The translation $t_b \in \mathbb{R}^3$ is the position of the distal joint expressed in the rotated coordinate system of the proximal joint. In effect, the output of forward kinematics is a set of 3D positions in world coordinates as well as rotations relative to the world coordinate axes for all joints.

Note that we assign relative joint rotations and translations to the *bones*, where the rotating joint of a bone is its proximal joint. At first sight, it may seem more natural to assign relative joint rotations and translations to the *joints*, which would lead to a “dual” definition of kinematic chains. However, such a definition would have two major disadvantages:

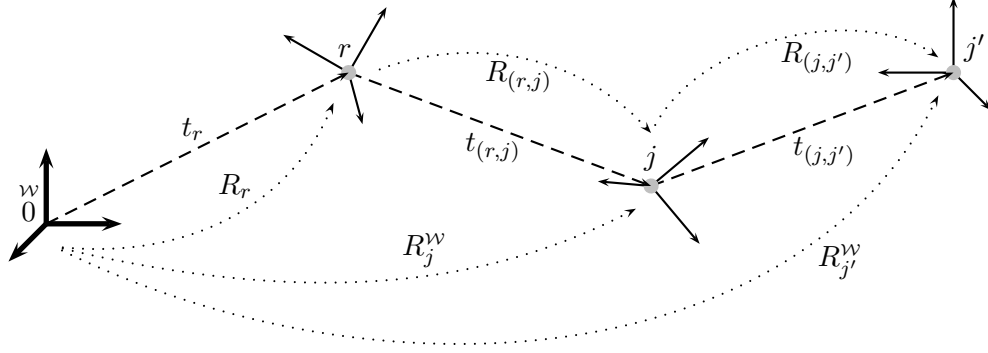


Figure 1.6. The principle of forward kinematics, illustrating Equations (1.6) and (1.7). The origin of the world coordinate system (left, bold coordinate axes) is marked as 0^w . The joints r , j , and j' are shown as gray dots along with their respective coordinate systems. Dotted arrows stand for rotations, while dashed arrows stand for translations (thus coinciding with the bones). All pairs (R, t) of corresponding rotations and translations describe Euclidean motions that act relative to the coordinate system at the starting point of the respective arrows, see Sect. 1.3.1.

1. The end effector joints would constitute a special case since we do not want any movement at or beyond the end effectors. We would end up with two sets of joints that would have to be treated differently: the rigid end effector joints and the movable inner joints. Note that in our original, bone-based definition, this behavior arises in a natural way since an end effector can never be the proximal joint of any bone.
2. Any joint with more than one incident bone would move these bones as a single rigid unit, since the bones are expressed in the same local joint coordinate system, which would move by a single joint rotation. Giving these bones the freedom of individual movement would require additional zero-length dummy bones, which is unnecessarily complicated. Nevertheless, such a joint-based definition of kinematic chains is used in the BVH mocap file format, see Sect. D.2.

In our bone-based definition of kinematic chains, the root joint constitutes a special case since it is not the distal joint of any bone—but this is more natural than the distinction between end effector joints and non-end effector joints since the root parameters R_r and t_r have a special meaning, which will be explained next.

Forward Kinematics. Given a kinematic chain $C = (J, r, B; R_r, t_r, (R_b)_{b \in B}, (t_b)_{b \in B})$, the starting point of forward kinematics is provided by the root rotation R_r and the root translation t_r , which establish the *root coordinate system* relative to the world system, see Fig. 1.6. Next, the bones that are incident to the root joint, i. e., all bones in the set $B_r := \{(r, j) \mid j \in J\} \cap B$, can be placed relative to the root coordinate system. For a bone $(r, j) \in B_r$, the world coordinates of the distal joint j are computed as in the local coordinate transform of (1.4):

$$t_j^w = \underbrace{R_r R_{(r,j)}}_{=: R_j^w} t_{(r,j)} + t_r, \quad (1.6)$$

where the rotation $R_{(r,j)}$ defines the rotation of the distal joint coordinate system against the root coordinate system, and the composite rotation R_j^w defines its rotation against the world

coordinate system. Note in particular that the root is simultaneously viewed as a proximal joint and a distal joint, i. e., the root can be thought of as a bone of length zero. This scheme is then continued in an analogous fashion for deeper levels of the hierarchy. For example, the world coordinates of the distal joint of a second-level bone $(j, j') \in B_j$, are computed as in (1.5):

$${}^w t_{j'} = R_j^w R_{(j,j')} t_{(j,j')} + {}^w t_j = R_r R_{(r,j)} \underbrace{R_{(j,j')}}_{=: R_j^w} t_{(j,j')} + R_r R_{(r,j)} t_{(r,j)} + t_r. \quad (1.7)$$

This bone placement scheme generalizes to Algorithm 1.1, FORWARDKINEMATICS, which computes world coordinates for all joints as well as joint rotations relative to the world coordinate system for a kinematic chain C .

Algorithm 1.1 FORWARDKINEMATICS

Input: Kinematic chain $C = (J, r, B; R_r, t_r, (R_b)_{b \in B}, (t_b)_{b \in B})$.

Output: $({}^w t_j)_{j \in J} \in \mathbb{R}^{3 \times |J|}$, $(R_j^w)_{j \in J} \in (\text{SO}_3)^{|J|}$,

${}^w t_j$: position of joint $j \in J$ in world coordinates,

R_j^w : rotation of the local coordinate system of joint $j \in J$ relative to the world coordinate system.

Procedure:

Starting at the root, use depth-first search or breadth-first search to traverse the kinematic chain while evaluating equations (1.8) and (1.9) for each $j \in J$ that is visited.

$$R_j^w := \begin{cases} R_r & \text{if } j = r, \\ R_{p(j)}^w R_{(p(j),j)} & \text{otherwise.} \end{cases} \quad (1.8)$$

$${}^w t_j := \begin{cases} t_r & \text{if } j = r, \\ R_j^w t_{(p(j),j)} + {}^w t_{p(j)} & \text{otherwise.} \end{cases} \quad (1.9)$$

For each $j \in J$, the tree structure and the traversal method ensure that $R_{p(j)}^w$ and ${}^w t_{p(j)}$ have been computed in a previous step. The entire procedure requires a total of $|B|$ matrix-matrix multiplications, $|B|$ matrix-vector multiplications, and $|B|$ vector additions.

1.3.3 Animated Skeletons and Motion Capture Data

Depending on the capturing hardware and the respective file format, raw motion capture data can either be a time-dependent sequence of 3D coordinates of certain marker points or it can be a time-dependent sequence of kinematic chain parameters, as we know from Sect. 1.2.1. In the former case, there is not much to be done to make the data accessible to our feature extraction (see Chap. 3), which works exclusively with sequences of 3D joint coordinates as defined in Def. 1.1. In the latter case, we must first convert the kinematic chain parameters to 3D joint coordinates. From this point forward, we will only focus on the 3D joint coordinates $({}^w t_j)_{j \in J}$, see Algorithm 1.1, since the rotations do not play a role for our purposes.

So far, we have considered fixed kinematic chains $C = (J, r, B; R_r, t_r, (R_b)_{b \in B}, (t_b)_{b \in B})$ yielding 3D joint coordinates $({}^w t_j)_{j \in J} \in \mathbb{R}^{3 \times |J|}$ via forward kinematics. If we let any of the

parameters $R_r, t_r, (R_b)_{b \in B}$, or $(t_b)_{b \in B}$ vary over time, we obtain a time-dependent sequence of kinematic chains, which in turn induces a time-dependent sequence of 3D joint coordinates. Observe from (1.6) and (1.7) that varying the parameters (R_r, t_r) of the root coordinate system changes the global placement of the entire kinematic chain in the world coordinate system since R_r and t_r always appear last in the transformation hierarchy. Variations of $(R_b)_{b \in B}$ induce rotations of the respective joints. Variations of the relative joint translations, $(t_b)_{b \in B}$, correspond to prismatic joints with translational degrees of freedom, or non-rigid bones. Recall that such behavior was ruled out in the introduction to Sect. 1.3, so the joint translations $(t_b)_{b \in B}$ will remain fixed.

The exact choice of $(t_b)_{b \in B}$ depends on the geometry of the actor's skeleton: the lengths of the translation vectors encode the lengths of the actor's bones, and the directions of the translation vectors define a standard pose that the skeleton assumes when all rotations are the identity. In practice, this standard pose is often chosen as in Fig. 1.4 or as a so-called T-pose, where both arms are stretched out to the side, see Fig. 1.3. Most available mocap systems require a semi-automatic calibration phase to determine appropriate values for $(t_b)_{b \in B}$, see Sect. 1.2.1.

In the case of fixed relative joint translations, we also speak of a *skeletal kinematic chain* or simply a *skeleton* and write

$$C = \underbrace{(J, r, B, (t_b)_{b \in B})}_{\text{skeleton}}; \underbrace{(R_r, t_r, (R_b)_{b \in B})}_{\text{DOF}}. \quad (1.10)$$

For the sake of clarity, we have regrouped the parameters of the kinematic chain into the *skeletal parameters* $J, r, B, (t_b)_{b \in B}$, and the *free parameters* $R_r, t_r, (R_b)_{b \in B}$, which are also known as the *degrees of freedom (DOF)* of the kinematic chain. For fixed skeletal parameters, the space of all possible free parameters is known as the *joint space* $\mathcal{J} := \text{SO}_3 \times \mathbb{R}^3 \times (\text{SO}_3)^{|B|}$.

Definition 1.3 (Animated skeleton)

Assuming a discretized time interval $[1 : T]$ as in Def. 1.1, an animated skeleton consists of fixed skeletal parameters $(J, r, B, (t_b)_{b \in B})$ together with a mapping

$$\begin{aligned} D_{\mathcal{J}} : [1 : T] &\rightarrow \mathcal{J} \\ t &\mapsto (R_r(t), t_r(t), (R_b(t))_{b \in B}), \end{aligned} \quad (1.11)$$

which encodes the evolution of the free parameters over time. By fixing a single joint, we obtain an angle trajectory as a curve $[1 : T] \rightarrow \text{SO}_3$, in analogy to joint trajectories as defined in Def. 1.1. \diamond

From an abstract point of view, the forward kinematics Algorithm 1.1 provides a mapping

$$f : \mathcal{J} \rightarrow \mathcal{P}, \quad (1.12)$$

which computes the 3D joint coordinates for a given set of free parameters. Applying forward kinematics for all $t \in [1 : T]$, we can then compute a motion capture data stream from an animated skeleton as $D := f \circ D_{\mathcal{J}}$,

$$\begin{aligned} D : [1 : T] &\rightarrow \mathcal{P} \\ t &\mapsto f(D_{\mathcal{J}}(t)) =: P(t). \end{aligned} \quad (1.13)$$

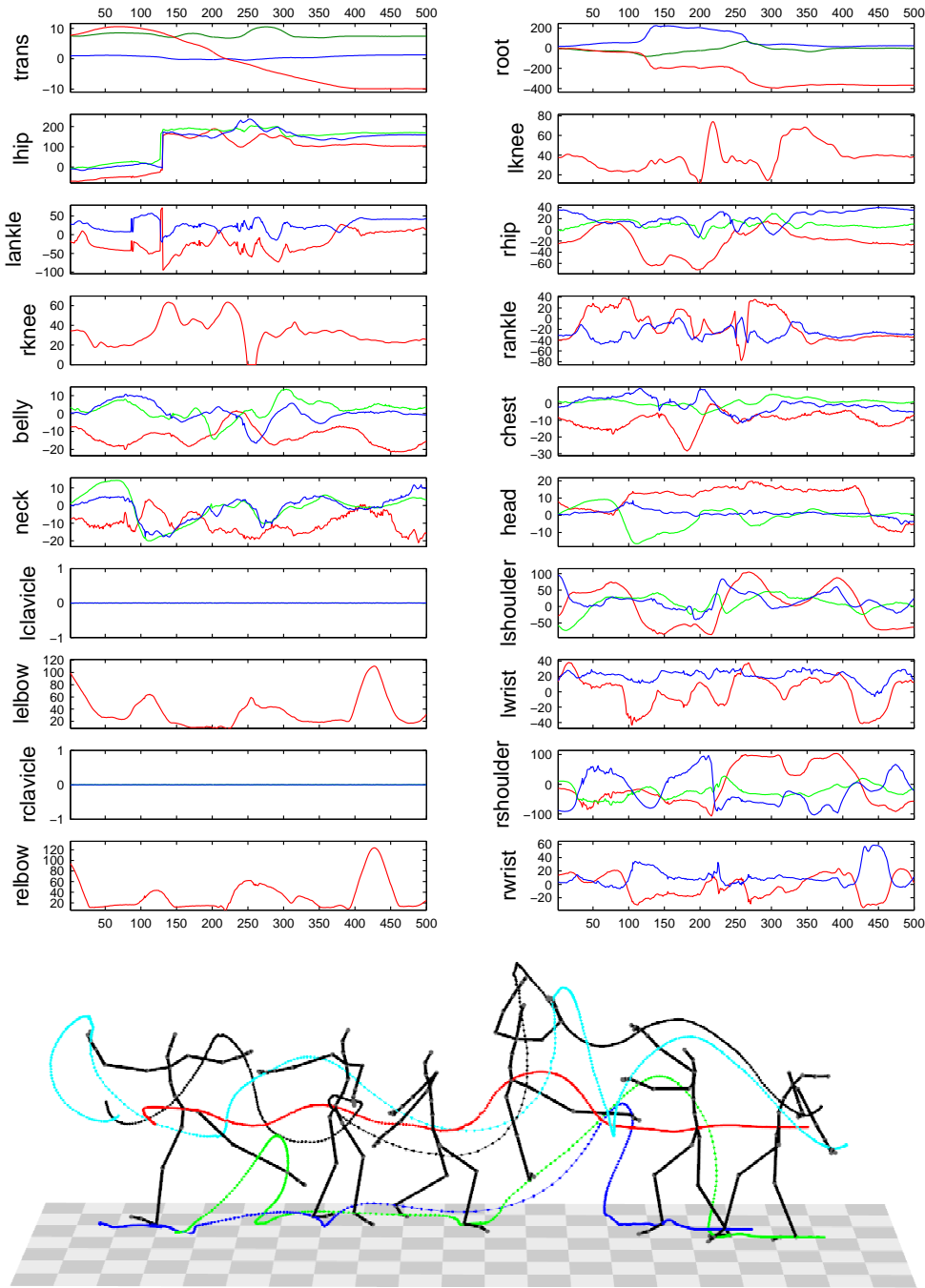


Figure 1.7. Top: Angle trajectories for a 500-frame ballet motion sampled at 120 Hz, adopted from the CMU mocap database [CMU03]. The motion is annotated as “attitude/arabesque, jeté en tourant, bending back” and comprises two 180° right turns, the second of which is jumped. The Euler angles for all inner joints, measured in degrees, are color-coded as red (x rotation), green (y rotation), and blue (z rotation). The upper-left subgraph shows the root translation, measured in multiples of 12.54 cm (a unit taken directly from the underlying mocap file format.) **Bottom:** The ballet motion visualized by selected poses along with the 3D trajectories for the joints ‘root’ (red), ‘r ankle’ (green), ‘l ankle’ (blue), ‘lwrist’ (black), ‘rwrist’ (cyan).

As an example, consider the ballet motion of Fig. 1.7, for which both the free parameters (root translation and Euler angle trajectories) as well as selected 3D joint trajectories are shown. A detailed account of rotational representations, including Euler angles, will be given in App. A.

1.3.4 Modeling Humans as Kinematic Chains: A Valid Simplification?

It is obvious from Fig. 1.4 that our kinematic chain does not incorporate all of the 206 bones that are typically found in the grown-up human body—on the contrary, we reduce the number of bones to 21, where we consider 19 joints comprising 1–3 rotational DOF. The total number of rotational DOF is 41. This simplification is justified by the fact that our focus lies on the analysis of large-scale, full-body motions. Our assumption is that most of the time, the semantics of such motions is sufficiently clear from our simplified skeletal model. Furthermore, we are deliberately masking out the details of human motion that would be required for biomechanical analysis or similar applications.

Beyond the number of bones and their geometric placement, skeletal kinematic chains differ from the anatomical and biomechanical truth in several other, more or less obvious ways, see [Zat98]. First, centers of rotation are not fixed with respect to the respective predecessor bones, and bones are neither perfectly rigid nor of fixed lengths. Hence, a skeletal kinematic chain model cannot capture all the subtleties of complex human motion. As an example, consider the motion of the human knee, which we model as a 1 DOF hinge joint. From the point of view of a biomechanist, the knee joint has 6 DOF, three of which are translational and three of which are rotational. The shoulder joint is an even more drastic example for the failure of our 3 DOF approximation; improved shoulder joint models are therefore used for tracking in computer vision [RKS⁺05], and even to enhance realism in animations [BPW93].

Only considering the skeleton ignores the fact that the human body mainly consists of soft tissue surrounding the bones. There are more elaborate ways of representing human motion such as the volumetric body model used by Green and Guan [GG04] to aid their tracking algorithm in the context of markerless action recognition from monocular video data. Here, an underlying kinematic chain is complemented by accurate surface, color, and reflectance models that are specified in cylindrical coordinates around the bones. Again, such detail is not required for our purpose since we exclusively work on 3D motion capture data for which the tracking task has already been carried out by means of an optical motion capture system.

However, motion capture data recorded by any marker-based motion capture system does contain errors that are due to the assumption of a pure rigid body system. Typically, one implicitly assumes that the optical markers are rigidly connected to the underlying bones, for which we have the skeletal model. This is not the case due to the actual soft-tissue structure of the human body: wobbling mass, particularly observed in obese subjects, as well as skin shifting are effects that can lead to significant displacements of the optical markers relative to the bones. In several comparative studies using bone-implanted optical markers in parallel with conventional skin-attached markers, Reinschmidt [Rei96] found from captured locomotion sequences that Euler angles at the knee and ankle joints tend to be systematically overestimated by optical markers due to skin shift. Typical magnitudes of deviation were 5°–10°. In terms of positional displacement, Lafortune et al. [LLL92] measured skin shifting of up to 7 cm at the knee joint.

In our own motion capture recording sessions, we observed a further, potentially even stronger source of errors. The black nylon suit that is used with the Vicon mocap system

both for reasons of optical contrast and as a fixation for the optical markers shifts relative to the body. For example, we found that the shoulder marker shifted by up to 10–12 cm during a cartwheel motion due to tension and wrinkling of the nylon suit. As a final point, converting the raw motion capture data, which consists of the optical markers' trajectories, to a skeleton-based representation may introduce significant errors due to heuristic fitting strategies as well as numerical errors.

In conclusion, we can state that the spatial resolution of optical mocap systems is often limited by the kinematic chain model and by certain factors that are inherent to the recording process with optical markers. Even though the markers can be tracked at sub-millimeter precision, the resulting animated skeleton will not be that accurate for many types of motions. Such inaccuracies are no problem for our coarse retrieval and classification tasks but may turn out to be troublesome for finer views on motion data.

Chapter 2

Similarity of Motions

Designing suitable similarity measures is crucial to content-based retrieval and classification of mocap data. Here, the main difficulty is that the underlying task of comparing two or more motions is inherently ill-defined. Answers to the corresponding questions,

“What is it that characterizes similar motions? How can similarity be quantified?”

are diverse and always depend on the intended application. For example, one may want to compare a walking and a jogging motion, see Fig. 2.1. At first sight, the trajectories and corresponding poses seem to be rather similar. However, looking into more detail, the two motions differ with respect to speed, relative timing, and important characteristics of the 3D trajectories. Depending on the application, one could consider these motions as similar because both of them are instances of locomotion, or as dissimilar because one motion is “walking” and the other motion is “jogging”.

Even in the comparison of two walking motions, large variations can be observed due to differences in direction and speed (Fig. 2.4 (a)), style (Figs. 2.4 (b), 2.2), absolute body size (Fig. 2.4 (a)), and many other parameters. A further factor is the identity of the performer, giving rise to intra-individual variations between motion performances (Fig. 2.2).

In summary, motions that are deemed *logically* similar (in the above example, both motions are instances of locomotion) need not be *numerically* similar (the motions exhibit signif-

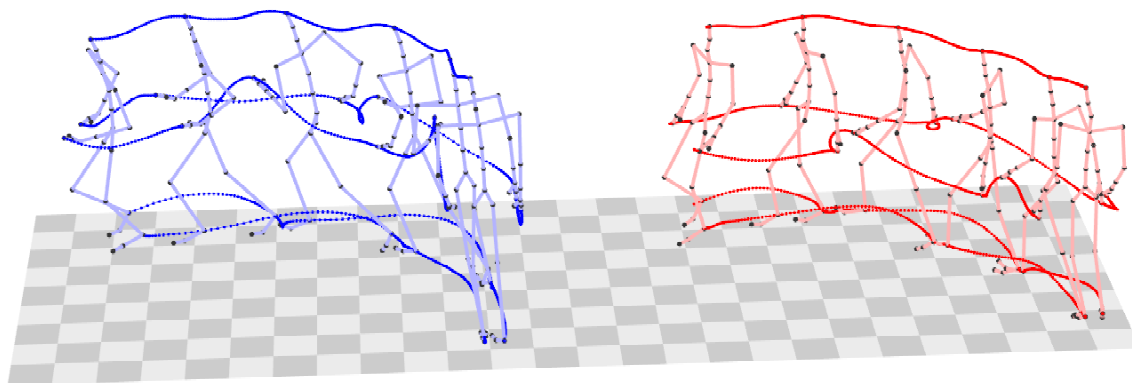


Figure 2.1. **Left:** four steps of a curved jogging motion. **Right:** four steps of a curved walking motion. Both motions start from a standing pose. The trajectories of the joints ‘headtop’, ‘rankle’, ‘lankle’, ‘rfingers’, and ‘lfingers’ are shown.

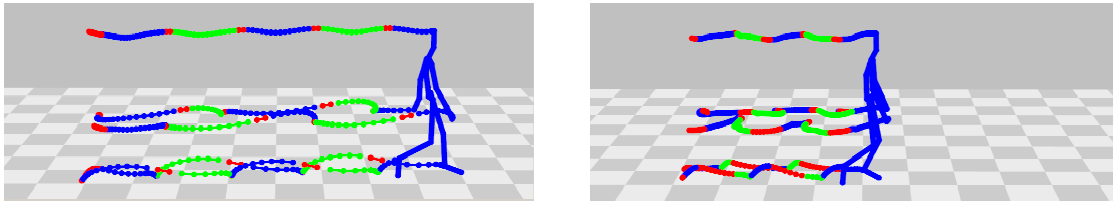


Figure 2.2. Two walking motions performed by different actors, in different speeds and styles. The figure shows the 3D trajectories for ‘headtop’, ‘rfingers’, ‘lfingers’, ‘rankle’, and ‘lankle’. Logically corresponding segments in the two motions are indicated by the same colors.

icant spatio-temporal variations). Conversely, we will also see examples of numerically similar motions that are not logically similar.

All of the above mentioned discriminative motion parameters such as walking direction, speed, or style belong to a set of more or less abstract *motion aspects* or *aspects of similarity*. Similarity measures for mocap data usually focus on some of these aspects while deliberately masking out others. In the case where an aspect is masked out, the similarity measure is said to be *invariant* to the respective aspect.

In the following, we will discuss several important aspects of similarity. We start in Sect. 2.1 with some remarks on human perception of motions. Then, we discuss certain coarse, global aspects of similarity in Sect. 2.2 and proceed to stylistic, emotional aspects in Sect. 2.3. The fundamental problem of assessing logical similarity by means of numerical comparison is discussed in Sect. 2.4. Common similarity measures that have been proposed in the literature are presented in Sect. 2.5, 2.6, and 2.7.

2.1 Human Perception of Motions

Many of our actions and much of our knowledge and skills depend on watching other humans or animals as they move. Therefore, human beings are experts at perceiving, understanding, and judging complex biological motion, see [Gol02, Chap. 8]. The field of *perception* is an important constituent of psychology that can provide hints as to what kind of information can be gained by visually observing human motion.

In the 1970s, Johansson [Joh75] developed the so-called *point-light display* to investigate human perception of walking motions. Here, several light dots attached to a moving person are recorded by a video camera in a darkened room. Test subjects watching such a video perceived a random point cloud as long as the lights were not moving. As soon as the recorded person started walking, the test subjects recognized a human walking motion (Fig. 2.3). This demonstrates both the capability of the human brain to extrapolate from extremely sparse data as well as the importance of the “structure from motion” concept, see [Gol02].

Moving from low-level motion perception towards human motion understanding, Kozlowski and Cutting [KC77] found that test subjects were able to infer the gender of a walking person from watching a corresponding point-light display. According to Runeson [RF81], it is even possible to accurately estimate the weight of lifted objects from point-light displays. Troje et al. [TWL05] found that human subjects performed very well at gait-based person identification from point-light displays. They demonstrated that humans tend to focus on kinematic parameters (gait frequency, amplitude) rather than on structural clues (absolute

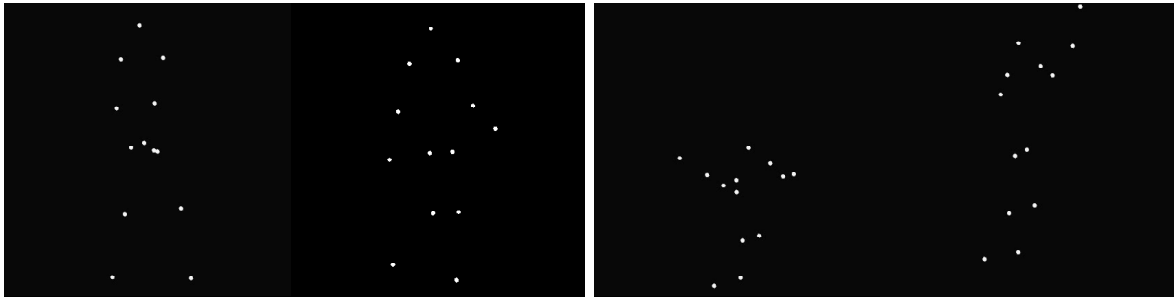


Figure 2.3. Point-light displays generated from mocap data, 45° frontolateral views onto the subject’s right-hand side. **Left:** two poses from a walking motion. **Right:** two poses from a jumping motion (arms swinging backwards and knees fully bent; maximum jumping height). Self-occlusions of the body have not been considered.

size, body proportions) to distinguish different persons. However, one is still far from developing a general theory of how humans perceive and process motions [Gol02, Chap. 8]. Here, concepts such as silhouette-based *temporal templates* [DB97] (not to be confused with *motion templates* as introduced in Chap. 5) have been investigated.

Regardless of the capabilities of the human perceptual system, our task is to develop automatic, computer-based methods for analyzing motion capture data. Typically, mocap data is richer in content than point-light displays: it is three-dimensional, usually comprises a larger number of markers, and provides structural information about how the individual points are connected among each other. Note in particular that one can synthesize point-light displays from mocap data [Tro02], see also Fig. 2.3. From these observations, one may conclude that mocap data is in principle rich enough for a computer system to extract similar high-level information as the human perceptual system is capable of extracting. In practice, of course, computers are far from performing such tasks as reliably. Nevertheless, there are specialized systems such as the mocap-based gender recognition system by Troje [Tro02] that provide reliable results at least for controlled data.

2.2 Global Aspects of Similarity

Usually, two motions will be regarded as similar if their 3D representations only differ by some global geometric transformation such as a translation or a rotation. To a certain extent, this basic notion of similarity can be formalized in terms of group theoretical concepts.

Given a group G acting on \mathbb{R}^3 such as the group of Euclidean motions, G induces a group action $G \times \mathcal{P} \rightarrow \mathcal{P}$ on the pose space \mathcal{P} via

$$(g, P) \mapsto gP, \quad (2.1)$$

where the j^{th} column of gP is defined as $(gP)^j := gP^j \in \mathbb{R}^3$. This group action on the pose space induces a group action on the set of motion capture data streams with time axis $[1 : T]$ by setting $(gD)(t) := gD(t)$ for $t \in [1 : T]$. Intuitively, the action of g on D amounts to a simultaneous transformation of the entire motion by applying g to every pose. Then, each mocap data stream $D : [1 : T] \rightarrow \mathcal{P}$ defines a G -orbit $GD := \{gD \mid g \in G\}$, thereby inducing an equivalence relation

$$D \equiv D' :\Leftrightarrow GD = GD' \quad (2.2)$$

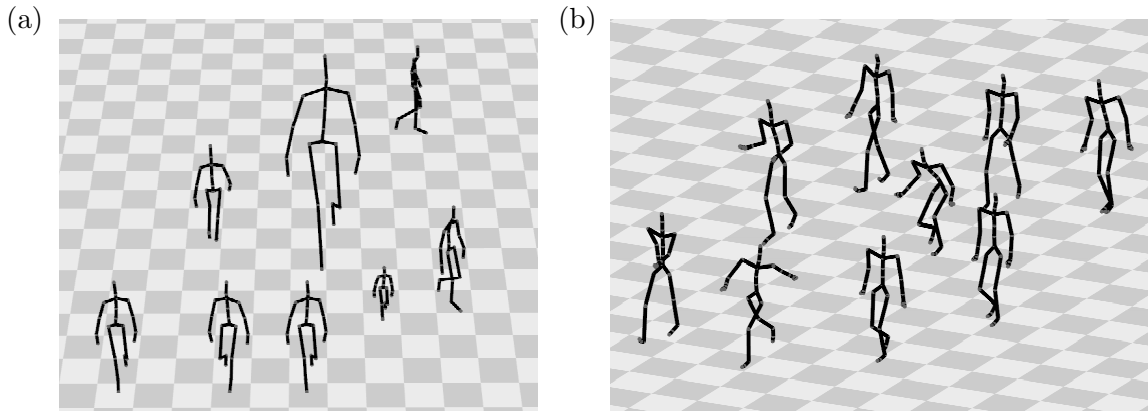


Figure 2.4. (a) Global transformations applied to a walking motion. (b) Different walking styles.

on the set of all mocap data streams with time axis $[1 : T]$. In other words, two mocap data streams D and D' are equivalent iff there is a group element $g \in G$ such that $gD = D'$. Of course, the resulting equivalence classes strongly depend on the group G .

One sensible choice for this *invariance group* G that will be encountered in Sect. 2.5.2 is a subgroup of the group of 3D Euclidean motions containing combinations of rotations about the (vertical) y axis with translations in the xz plane. Intuitively, two motions are equivalent under this invariance group if they only differ by their absolute location in the xz plane and a global rotation about the y axis. The invariance group could also be extended to include uniform scalings of \mathbb{R}^3 as well as reflections. Examples of different walking motions that are equivalent with respect to such an invariance group are shown in Fig. 2.4 (a).

Note that the strict requirement of identity in (2.2) leads to a boolean concept of similarity, where motions are either entirely similar or entirely dissimilar, without any *ranking* in between. This requirement can be softened to some extent, leading to fault-tolerant concepts of similarity [CK04].

So far, the temporal aspect has been disregarded in this formulation: only mocap data streams with the common time axis $[1 : T]$ can be compared. Instead, one could admit global temporal scalings or even local stretching and contraction of the time axis. This leads to the well-known method of *dynamic time warping*, enabling the comparison of data streams that differ with respect to global and local spatio-temporal deformations, albeit at high computational cost, see Sect. 2.4.1 and 2.7. In Chap. 3 and 4, we will present our feature extraction and retrieval methods, which provide new ways of efficiently comparing motions irrespective of spatial and temporal deformations.

2.3 Content vs. Style

More complex are variations that are due to different motion styles, see Figure 2.4 (b). For example, walking motions may differ by performance (e.g., limping, tiptoeing, or marching), by emotional expression or mood (e.g., “cheerful walking”, “furious walking”, “shy walking”), and by the complex individual characteristics determined by the person performing the motion. The abstract concept of *motion style* appears in the literature in various forms and is usually contrasted by some notion of *motion content* that is related to the semantics

of the motion. Applications of motion retrieval and classification typically aim at identifying related motions by content irrespective of motion style, see Sect. 2.4. Notable exceptions are applications such as person and gender recognition, where certain aspects of motion style are the decisive factor for asserting similarity. In the following, we give an overview of how motion style and motion content are treated in the literature.

In the context of biometric gait analysis, Lee and Elgammal [LE04] define motion style as the time-invariant, personalized aspects of gait, whereas they view motion content as a time-dependent aspect representing different body poses during the gait cycle. Similarly, Davis and Gao [DG03] view motions as depending on style, pose, and time. In their experiments, they use PCA on expert-labeled training data to derive those factors (essentially linear combinations of joint trajectories) that best explain differences in style. Rose et al. [RCB98] group several example motions that only differ by style into *verb* classes, each of which corresponds to a certain motion content. They synthesize new motions from these verb classes by suitable interpolation techniques, where the user can control interpolation parameters for each verb. These parameters are referred to as *adverbs* controlling the style of the verbs. To synthesize motions in different styles, Brand and Hertzmann [BH00] use example motions to train so-called *style machines* that are based on hidden Markov models (HMMs). Here, motion style is captured in certain parameters of the style machine such as average state dwell times and emission probability distributions for each state. On the other hand, motion content is encoded as the most likely state sequence of the style machine. Hsu et al. [HPP05] propose a system for *style translation* that is capable of changing motions performed in a specific input style into new motions with the same content but a different output style. The characteristics of the input and output styles are learned from example data and are abstractly encoded in a linear dynamic system. A physically-based approach to grasping the stylistic characteristics of a motion performance is proposed by Liu et al. [LHP05]. They use a complex physical model of the human body including bones, muscles, and tendons, the biomechanical properties of which (elasticity, stiffness, muscle activation preferences) can be learned from training data to achieve different motion styles in a synthesis step. Troje [Tro02] trains linear PCA classifiers to recognize the gender of a person from recorded gait sequences, where the “gender” attribute seems to be located in the first three principal components of a suitable motion representation. Using a Fourier expansion of 3D locomotion data, Unuma et al. [UAT95] identify certain *emotional* or *mood* aspects of locomotion style (for instance, “tired”, “brisk”, “normal”) as gain factors for certain frequency bands.

Pullen and Bregler [PB02] also use a frequency decomposition of motion data, but their aim is not to pinpoint certain parameters that describe specific styles. Instead, they try to extract those details of the data that account for the natural look of captured motion by means of multiresolution analysis (MRA) on mocap data [BW95]. These details are found in certain high-frequency bands of the MRA hierarchy and are referred to as *motion texture* in analogy to the texture concept in computer graphics, where realistic surfaces are rendered with texture mapping. The term ‘motion texture’ is also used by Li et al. [LWS02] in the context of motion synthesis, but their concept is in no way related to the signal processing approach of Pullen and Bregler [PB02]. In their parlance, motion textures are generative statistical models describing an entire class of motion clips. Similar to style machines [BH00], these models consist of a set of *motion textons* together with transition probabilities encoding typical orders in which the motion textons can be traversed. Each motion texton is a linear dynamic system (see also Hsu et al. [HPP05]) that specializes in generating certain subclips of the modeled motion. Parameter tuning at the texton level then allows for manipulating

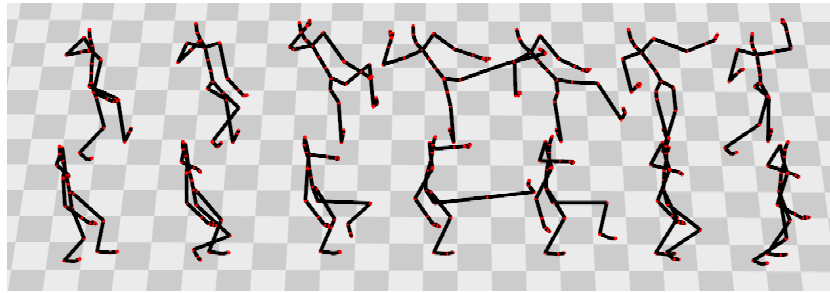


Figure 2.5. Top row: a side kick. **Bottom row:** a frontal kick (2.3 seconds each.) The motions are illustrated by selected poses that have been pulled apart in horizontal direction for a better visualization.

stylistic details.

Inspired by the performing arts literature, Neff and Fiume [NF04, NF05] explore the aspect of *expressiveness* in synthesized motions. Their system enables the user to describe motion content in a high-level scripting language. The content can be modified globally and locally by applying procedural *character sketches* and *properties*, which implement expressive aspects such as “energetic”, “dejected”, or “old man”.

Returning to the walking example of Figure 2.4 (b), we are faced with the question of how a walking motion can be characterized and recognized irrespective of motion style or motion texture. Video-based motion recognition systems such as [Bre97, GG04] tackle this problem by using hierarchical HMMs to model the motion content. The lower levels of the hierarchy comprise certain HMM building blocks representing fundamental components of full-body human motion such as “turning” or “raising an arm”. In analogy to *phonemes* in speech recognition, these basic units are called *dynemes* by Green and Guan [GG04] or *movemes* by Bregler [Bre97]. Dynemes/movemes and higher-level aggregations of these building blocks are capable of absorbing some of the motion variations that distinguish different executions of a motion.

2.4 Logical vs. Numerical Similarity

Our aim is to design similarity measures that consider two motions as similar if they represent variations of the same action or sequence of actions. This notion is referred to as *logical similarity*, see [KG04]. The “variations of an action or sequence of actions” typically concern both the spatial and the temporal domain. In other words, we try to focus on motion content while masking out motion style or texture. For example, the two walking motions shown in Fig. 2.2 can be regarded as similar from a logical point of view even though they differ considerably w. r. t. speed and style. As a further example, consider the two kicking motions shown in Fig. 2.5: the kick in the top row is performed after the hip has been rotated to the left, whereas the kick in the bottom row is performed with the hip facing forwards. However, from a logical point of view, both motions are kicks that are directed to the right. Regarding the two jumps shown in Fig. 2.6, the motion in the top row is much more energetic than the motion in the bottom row, which can be seen from the different arm swings and the bending of the knees. Yet both motions are forward jumps. Using the terminology of information retrieval, such a situation may lead to a *false negative*: logical similarity that cannot be

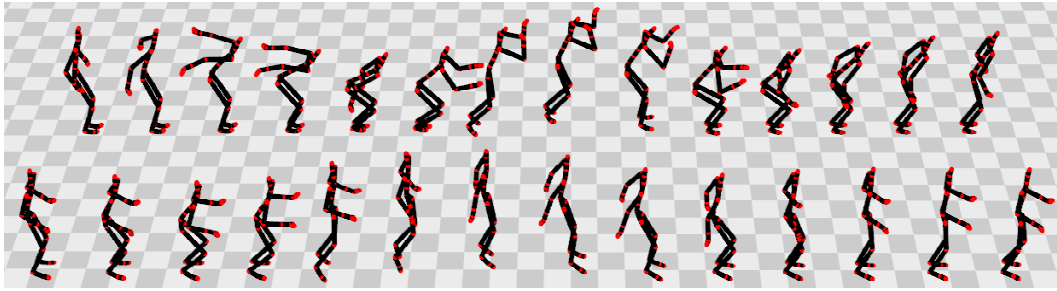


Figure 2.6. Top row: a forceful jump. Bottom row: a weak jump (1.4 seconds each.)

revealed by numerical comparison of the 3D coordinates or joint angle parameters.

Conversely, numerical similarity does not necessarily imply logical similarity. Often, only minor nuances or partial aspects of a motion account for logical differences. Think of the motions “standing on a spot” compared to “standing accompanied by weak waving with one hand”: such inconspicuous, but decisive details are difficult to pick up for a full-body similarity measure unless the focus of the similarity measure is primarily on the motion of the hands. As a further example, consider the difference between walking and jogging as illustrated in Fig. 2.1. These motions may of course be distinguished by their absolute speed. Yet, the overall shape of most joints’ trajectories is very similar in both motions. A better indicator would be the occurrence of simultaneous air phases for both feet, which is a discriminative feature of jogging motions, see also Sect. 3.5. To carry this effect to the extremes, think of the difference between “placing an object on a table” and “picking up an object from a table” [KG04]. Even for a human, these motions are very hard to distinguish without any information about whether the hand is gripping an object or not, see [KP06]. This may lead to a *false positive*: logical similarity that is falsely asserted based on numerical comparison of 3D coordinates or joint angle parameters.

Bridging the *semantic gap* between logical similarity as perceived by humans and numerical similarity as can be quantified by a computer is one of the fundamental problems addressed by this thesis. The following subsections discuss some of the aspects of similarity that are particularly problematic in this context.

2.4.1 Spatio-Temporal Variations

In comparing two motions, one can often identify a common temporal structure such as the periodic gait cycle of the walking and the running motions shown in Fig. 2.1. The six poses shown for each of the motions roughly correspond to each other in that they belong to the same phase of the gait cycle: both motions start with a standing pose, then the step sequence “right-left-right-left” is performed. However, the gait cycles are out of phase, and the relative phase shift may even change over time, just as the walking or jogging speed may change from step to step. We refer to such distortions as *temporal variations*. Also, the poses belonging to corresponding phases exhibit *spatial variations* when comparing walking and jogging: in the jogging motion, the upper body leans forwards and the elbows are bent, while in the walking motion, the posture is upright and the elbows are stretched. In combination, we refer to such distortions as *spatio-temporal variations*.

Fig. 2.7 (a) demonstrates how minor local spatial variations (here: slightly turning to the

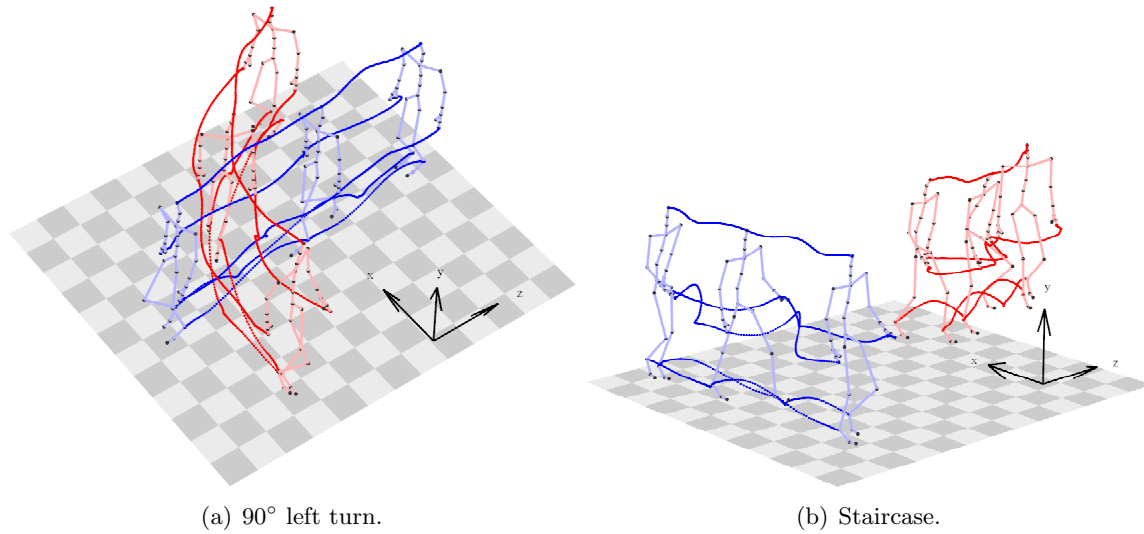


Figure 2.7. (a) Two walking motions comprising four steps (2.7 seconds). Both characters initially look down the $-z$ axis (approximately), then the red character turns to the left by almost 90° and ends up looking down the $-x$ axis. Simultaneously, the blue character walks in a straight line parallel to the $-z$ axis. (b) Two walking motions comprising three steps (1.8 seconds). The red character is climbing a staircase.

left from step to step) can accumulate into major global deformations, yielding significantly different trajectories. Also, interaction with the environment as shown in Fig. 2.7 (b) can lead to spatio-temporal variations—the trajectories move upwards, following the (invisible) staircase. Note that spatio-temporal variations are not restricted to cyclic motions. They may occur in arbitrary, non-periodic motions such as jumping (Fig. 2.6) or kicking (Fig. 2.5).

It has been mentioned that the technique of dynamic time warping (DTW) can handle temporal variations by suitably stretching or contracting the time axis to align corresponding poses. Here, correspondence is asserted by means of a *local similarity measure* measuring the similarity between isolated poses or between small temporal neighborhoods of poses, see Sect. 2.5. To a certain extent, local similarity measures can absorb spatial variations. Altogether, DTW is capable of comparing motions that differ by spatio-temporal variations.

2.4.2 Partial Similarity

Are the two instances of “rotating both arms forwards” as shown in Fig. 2.8 similar? In other words, should the superimposed walking motion (blue character) have an influence on the comparison? Looking at the corresponding 3D representations, the circular trajectory of the hand is dragged out into a cycloid. Hence, the motions could be considered as dissimilar since a cycloid and a circle are very different curves. The influence of the forward motion can be eliminated by moving the root coordinate system of each pose into the world coordinate system, which would transform the cycloid trajectory back into a circle. But even then, the legs of the two characters would move in a different way.

The underlying problem is that of *partial similarity* or, more generally, *relevance*. Only certain parts of the body (the arms) move in a similar way, while other parts (the legs) move differently. Automatically detecting which parts of the body contain the relevant aspects is a

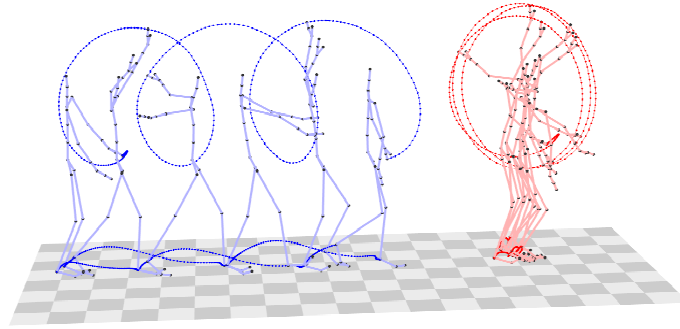


Figure 2.8. Three repetitions of “rotating both arms forwards”. The blue character is walking while rotating the arms (2.7 seconds), whereas the red character is standing on one spot while rotating the arms (2.3 seconds). The trajectories of the joints ‘r ankle’, ‘l ankle’, and ‘l fingers’ are shown.

difficult problem that is unsolved in the general case. The new concept of *motion templates* as presented in Chap. 5 takes a step towards automatic detection of relevant motion aspects. In general, however, some kind of manual input will be required to direct the “attention” of the similarity measure to the relevant aspects. For example, a user of a motion retrieval system could select the body parts that should be regarded in the comparison.

2.4.3 Similarity of Subsequences

Partial similarity as introduced above refers to the spatial aspects of motions in that certain parts of the body or motion-derived features may behave similarly or differently, regardless of any temporal aspects. The analogous concept for the temporal domain is that of *similarity of subsequences*. In a typical retrieval scenario, one is given a long *database motion*, D , that is to be compared with a short *query motion*, Q . The question is whether there is a subsequence of D that is similar to Q , and not whether D is similar to Q in its entirety. Hence, one needs appropriate mechanisms for identifying similar subsequences. One such method is that of *subsequence DTW*, see [RJ93]. By contrast, one of our proposed retrieval techniques (Chap. 4) uses a generalized form of exact full-text retrieval to efficiently identify matching motion subsequences.

2.4.4 Noise and Artifacts

Noise is a further factor that may interfere with a similarity measure for motion clips. Mocap data may contain high frequency noise as well as undesirable artifacts such as sudden “flips” of a joint or systematic distortions due to wobbling mass, skin shift, or inadequacy of the kinematic chain model, see Sect. 1.3.4. For example, consider the toe trajectory shown in the ballet motion of Fig. 2.9, where the noise shows as extremely irregular sample spacing. Such noise is usually due to adverse recording conditions, improper setup or calibration, or data conversion errors. On the left hand side of Fig. 2.9, there is a discontinuity in the trajectory, which results from a 3-frame flip of the hip joint. Such flips either result from confusions of markers in the tracking process or from the mapping of marker data onto an abstract skeletal model. Both high-frequency noise and discontinuities can also be observed in the corresponding angle trajectories shown in Fig. 1.7. Ren et al. [RPE⁺05] have developed

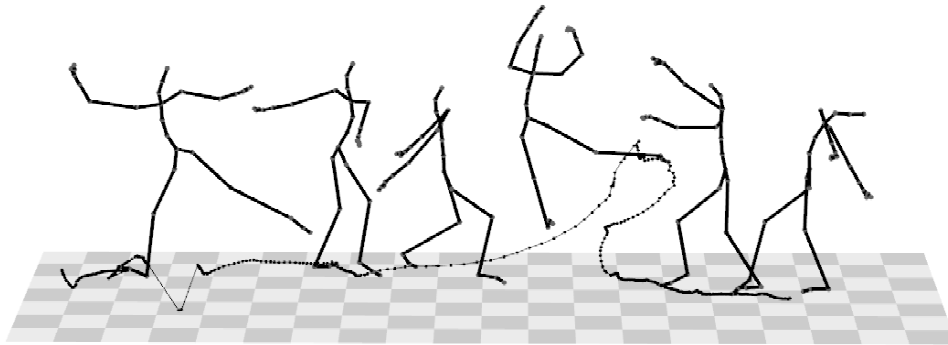


Figure 2.9. The ballet motion of Fig. 1.7. Only the noisy trajectory of the joint ‘ltoes’ is shown.

automatic methods for detecting “unnatural” movements in order to find noisy clips or clips containing artifacts within a mocap database.

In practice, noisy mocap data such as the clip of Fig. 2.9 could not be used in the production of a computer generated movie. One would first apply filtering and semi-automatic cleaning techniques to remove the noise and the artifacts. Some typical motion filtering techniques are described and compared in App. B. Also compare the noisy toe trajectory of Fig. 2.9 to the filtered toe trajectories shown in Fig. B.4. Noise and artifacts are also a problem in markerless, video-based mocap systems such as the system by Rosenhahn et al. [RKS⁺05]. In view of such scenarios, it is important to design noise-tolerant similarity measures for the comparison of mocap data. The use of *qualitative, relational* features opposed to *quantitative, numerical* features is a possible strategy to overcome some of the above mentioned difficulties, see Chap. 3.

2.5 Local Similarity Measures

Motions can either be viewed as time-synchronized bundles of 3D trajectories (*horizontal* view, fixing a joint) or as sequences of poses (*vertical* view, fixing a frame number), cf. Def. 1.1. The vertical view is most often adopted in the construction of *local similarity measures*, where individual poses are compared to each other. In a further step, such local similarity measures are then incorporated into *global similarity measures* that take the temporal aspect of motions into account, see Sect. 2.7.

At this point, it is interesting to note that even a single, expressive pose may convey a lot of information about the motion context in which it is embedded, see also [NF04]. As a simple example, consider the kicking motions of Fig. 2.5. The plain fact that the right leg is raised with a stretched knee during the middle phase of the motion is sufficient to distinguish “kicking” from many other motion classes. In general, it is often easier or more natural to construct a *dissimilarity* measure than a similarity measure, i. e., a function $c : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}_{\geq 0}$ that assumes the value zero for identical poses and positive values for dissimilar poses. Such functions are often referred to as *cost measures*. It is usually possible to modify a cost measure in such a way that it assumes larger values for higher degrees of similarity and vice versa, thereby assuming the role of a *score function*. When appropriate, we therefore use the terms “cost measure”, “dissimilarity measure”, “distance measure”, “score function”, and “similarity measure” interchangeably.

In the following, we exemplarily discuss two local distance measures that are proposed in the literature. We also performed some experiments with these distance measures, which are described in Sect. 2.8. Other local distance measures, some of which are based on joint velocities and accelerations, can be found in [AFO03, FF05, LCR⁺02]. For an overview and comparative evaluation, see also [GY05].

2.5.1 Quaternion-Based Pose Distance

We have discussed in Sect. 1.3 how a character's pose can be encoded by means of an absolute translation and rotation for the root together with relative joint rotations for each of the bones. One possible way of encoding the joint rotations is to use unit quaternions as described in Sect. A.3. It turns out that a quaternion-based motion representation is well-suited for the comparison of motions for the following reasons:

- Relative joint rotations are invariant to the group of 3D Euclidean motions.
- The geodesic distance on the set of unit quaternions, which forms the 3-sphere S^3 , provides a natural distance measure for rotations.

However, there is a major drawback to comparing motions based on relative joint rotations, as is also noted in [KGP02]: some of the rotations have a much greater overall effect on the character's pose than others. For example, small rotations in the shoulder joint can lead to large changes in the position of the entire arm. Conversely, even large rotations in the wrist joint do not have much influence on the overall pose. To make things worse, it is not possible to fully compensate for this effect by introducing (constant) joint weights since the influence of a rotation depends on the current pose. For example, the influence of the shoulder rotation on the overall pose is larger with a stretched elbow than with a bent elbow.

Given two animated skeletons (cf. Sect. 1.3.3)

$$D_{\mathcal{J}}(t) = (R_r(t), t_r(t), (R_b(t))_{b \in B}) \text{ and } D'_{\mathcal{J}}(t) = (R'_r(t), t'_r(t), (R'_b(t))_{b \in B}) \quad (2.3)$$

with common skeletal parameters $(J, r, B, (t_b)_{b \in B})$, we want to measure the distance between two frames $D_{\mathcal{J}}(n)$ and $D'_{\mathcal{J}}(m)$. To this end, we consider the individual distances between corresponding joint rotations $R_b(n)$ and $R'_b(m)$ for $b \in B$. Let q_b and q'_b be the unit quaternions describing the relative joint rotations $R_b(n)$ and $R'_b(m)$, respectively. Similar to Johnson [Joh03] and Lee et al. [LCR⁺02], we define the *quaternion-based pose distance* between the two frames as the total weighted quaternion distance

$$c^{\text{quat}}(n, m) := c^{\text{quat}}(D_{\mathcal{J}}(n), D'_{\mathcal{J}}(m)) = \sum_{b \in B} w_b \cdot \frac{2}{\pi} \cdot \arccos |\langle q_b, q'_b \rangle|, \quad (2.4)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in \mathbb{R}^4 and the w_b are suitable weights with $\sum_{b \in B} w_b = 1$. The terms

$$c^b(q_b, q'_b) := \frac{2}{\pi} \cdot \arccos |\langle q_b, q'_b \rangle| \quad (2.5)$$

measure distances between unit quaternions and can be interpreted as follows. The scalar product $\langle q_b, q'_b \rangle$ gives the cosine of the angle $\varphi \in [0, \pi]$ enclosed between q_b and q'_b , hence $\varphi = \arccos \langle q_b, q'_b \rangle$. This is the geodesic distance, i. e., the length of the shortest connecting path between the points q_b and q'_b on the four-dimensional unit sphere S^3 , see also App. B.

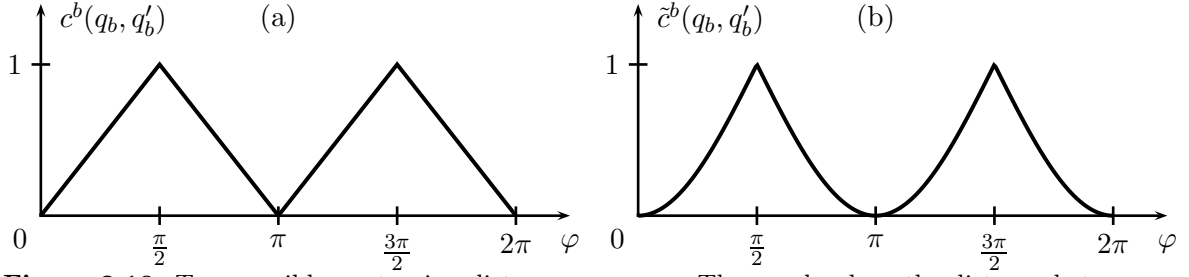


Figure 2.10. Two possible quaternion distance measures. The graphs show the distance between q_b and q'_b plotted against the enclosed angle $\varphi \in [0, \pi]$. The geodesic distance measure c^b is shown in (a), while the cosine measure \tilde{c}^b is shown in (b).

What we are actually looking for is the geodesic distance on $\mathbb{R}P^3 = S^3/\{-1, 1\}$, which is the set of unit quaternions where antipodal points are identified with each other. To see this, recall that the quaternion parametrization defines a double cover of SO_3 where antipodal points in S^3 describe the same rotation, see Sect. A.3. As the following considerations show, taking the absolute value of the scalar product in (2.5) simulates this identification of antipodal points on S^3 .

Starting from the situation where $q_b = q'_b$, we have $\varphi = c^b(q_b, q'_b) = 0$, see Fig. 2.10 (a). Now we let q_b move away from q'_b with constant velocity, which leads to linearly increasing values of φ . As soon as $\varphi = \frac{\pi}{2}$, the two unit quaternions are perpendicular, and $c^b(q_b, q'_b) = 1$ due to the (optional) normalization factor of $\frac{2}{\pi}$. When q'_b moves even further away from q_b than $\varphi = \frac{\pi}{2}$, the value of $c^b(q_b, q'_b)$ decreases linearly and reaches zero at $\varphi = \pi$. This is because q'_b is now approaching and eventually reaching $-q_b$, which is identified with q_b . Moving on to the case $\varphi > \pi$, the sequence repeats itself: q'_b moves away from $-q_b$ until $\varphi = \frac{3\pi}{2}$ and then returns to q_b , where $\varphi = 2\pi \equiv 0$.

It is noted in [Sho85] that the geodesic distance on $\mathbb{R}P^3$ is equivalent to the angular distance on SO_3 . As a consequence, moving with constant speed along the geodesic on $\mathbb{R}P^3$ from q_b to q'_b corresponds to a rotation with constant angular velocity that transforms the orientation encoded by q_b into the orientation encoded by q'_b , see also Sect. A.3.

Another possible quaternion distance is given by the cosine measure

$$\tilde{c}^b(q_b, q'_b) := 1 - |\langle q_b, q'_b \rangle|, \quad (2.6)$$

which does not increase and decrease linearly. Instead, it behaves like $1 - |\cos \varphi|$, see Fig. 2.10 (b). This penalizes larger deviations more than smaller deviations.

To some extent, the weights w_b may be used to compensate for the above mentioned effect that different joints have different amounts of influence on the overall 3D pose. For example, one could give some of the proximal joints (belly, shoulders, hips) twice as much weight as some of the more distal joints (neck, elbows, knees), and less weight to the wrists and the ankles. This also allows the designer or user of a similarity measure to account for partial similarity as discussed in Sect. 2.4.2.

2.5.2 3D Point Cloud Distance

In their work on *motion graphs*, Kovar and Gleicher [KGP02] propose a technique to identify motion subclips within a given motion clip D that can be concatenated without visible artifacts at the transition points. They consider a transition from frame n to frame m as suitable

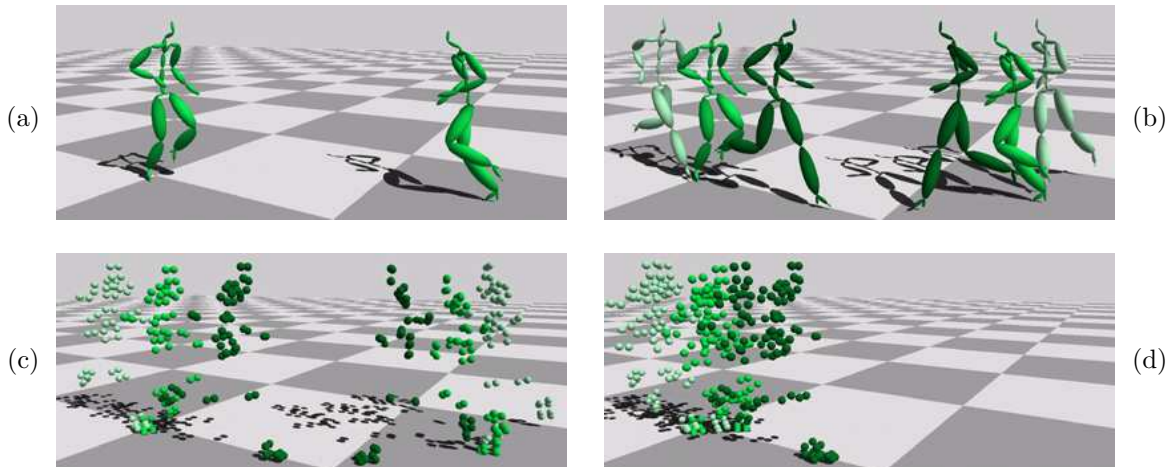


Figure 2.11. Principle of the 3D point cloud distance, adopted from [KG03]. (a) The two poses that are to be compared. (b) Inclusion of a temporal context for each of the poses. (c) Conversion to point clouds. (d) Optimal alignment of point clouds to determine pose distance as sum of distances of corresponding points.

if the corresponding poses $D(n)$ and $D(m)$ are similar with respect to a local distance measure that is based on a comparison of certain pose-driven 3D point clouds.

The principle of their distance measure is explained in Fig. 2.11. Here, the goal is to compare the two poses from a running sequence that are shown in Fig. 2.11 (a). Since we want to detect smoothly blendable motion subclips, a temporal context around both frames has to be incorporated in the comparison, as depicted in Fig. 2.11 (b). This is required because the notion of smoothness implicitly depends on temporal derivatives, which in turn depend on a local temporal context. Next, the two short pose sequences are converted to 3D point clouds, see Fig. 2.11 (c). Finally, an optimal alignment of the two point clouds is computed, and the total residual distance between corresponding points constitutes the *3D point cloud distance* between frames n and m .

The authors of [KGP02] conjecture that measuring *perceptual* similarity between poses would require to obtain these point clouds from a downsampled version of the 3D mesh representing the character’s skin. Here, they argue that the skeleton-driven skin is all that is seen by an observer, and that the skeleton is only a means to an end. Such an approach is realistic in the context of professional animation systems, which provide that kind of skinning and mesh data. However, for the sake of simplicity, we will use the relatively sparse 3D joint coordinates of a skeleton-based motion capture representation or the raw 3D marker data. The latter type of data is even closer to the downsampled mesh data proposed by Kovar and Gleicher, since the optical markers are attached to the surface of the actor’s body. Our experiments show that the type of point cloud data has very little effect on the results.

We will now formally define the 3D point cloud distance. Let D, D' be motion capture data streams with common joint set J . In comparing the pose $D(n)$ to the pose $D'(m)$, we view both poses in the context of the ρ preceding frames and the ρ subsequent frames, yielding the $2\rho + 1$ frame numbers $[n - \rho : n + \rho]$ and $[m - \rho : m + \rho]$, respectively. For D , the 3D points corresponding to these poses form an ordered point cloud $(D^j(i))_{i \in [n - \rho, n + \rho], j \in J}$ containing $K := |J|(2\rho + 1)$ points in \mathbb{R}^3 . For short, we denote this point cloud by $H := (p_k)_{k \in [1:K]}$.

Analogously, we define the point cloud H' for the pose $D'(m)$. As with all sliding window techniques, the boundary cases require special attention. Here, the motions need to be padded by ρ additional frames at the beginning and at the end. We employ symmetric padding using parts of a time-reversed version of the original data streams, see Eqn. (3.24).

Having converted both poses into their point cloud representations, yielding $H = (p_k)_{k \in [1:K]}$ for $D(n)$ and $H' = (p'_k)_{k \in [1:K]}$ for $D'(m)$, where $p_k = (x_k, y_k, z_k)^\top \in \mathbb{R}^3$ and $p'_k = (x'_k, y'_k, z'_k)^\top \in \mathbb{R}^3$, we align the two point clouds: H' is suitably rotated about the y axis by a rotation $R_y(\alpha)$ and then shifted in the xz plane by an offset vector $p := (x, 0, z)^\top \in \mathbb{R}^3$. Symbolically, we apply the following transformation to all p'_k , $k \in [1:K]$:

$$R_y(\alpha)p'_k + p = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \begin{pmatrix} x'_k \\ y'_k \\ z'_k \end{pmatrix} + \begin{pmatrix} x \\ 0 \\ z \end{pmatrix}. \quad (2.7)$$

The parameters $\alpha \in \mathbb{R}/2\pi\mathbb{Z}$, $x \in \mathbb{R}$, and $z \in \mathbb{R}$ are chosen so as to minimize the error function

$$f(\alpha, x, z) := \sum_{k=1}^K w_k \|p_k - (R_y(\alpha)p'_k + p)\|^2, \quad (2.8)$$

where the $w_k \in \mathbb{R}_{\geq 0}$ are suitable weights with $\sum_{k=1}^K w_k = 1$. The value of f expresses the weighted total squared Euclidean distance between corresponding points in H and the transformed version of H' . Then, the *3D point cloud distance* $c^{3D}(n, m)$ between frames n and m is the residual alignment error resulting from an optimal alignment of H and H' with respect to the error function f :

$$c^{3D}(n, m) := c^{3D}(D(n), D'(m)) = \min_{\alpha, x, z} \sum_{k=1}^K w_k \|p_k - (R_y(\alpha)p'_k + p)\|^2. \quad (2.9)$$

Finding such an optimal alignment constitutes a restricted version of a so-called *Procrustes problem*. Unrestricted Procrustes problems typically involve 3D Euclidean motions with six DOF, opposed to our three DOF alignment, see [Fio01]. The following theorem gives explicit formulas for the optimal parameters. A proof is given in App. C:

Theorem 2.1 *The optimal parameters minimizing (2.8) are*

$$\hat{\alpha} = \arctan \left(\frac{\sum_{k=1}^K w_k (x_k z'_k - z_k x'_k) - (\bar{x} \bar{z}' - \bar{z} \bar{x}')}{\sum_{k=1}^K w_k (x_k x'_k + z_k z'_k) - (\bar{x} \bar{x}' + \bar{z} \bar{z}')} \right) \quad (2.10)$$

$$\hat{x} = \bar{x} - \bar{x}' \cos \hat{\alpha} - \bar{z}' \sin \hat{\alpha} \quad (2.11)$$

$$\hat{z} = \bar{z} + \bar{x}' \sin \hat{\alpha} - \bar{z}' \cos \hat{\alpha}, \quad (2.12)$$

where $\bar{x} := \sum_{k=1}^K w_k x_k$ is the x component of the centroid of H , and the other barred terms are defined similarly.

An equivalent way of stating (2.11) and (2.12), see also Lemma C.1, is

$$\begin{pmatrix} \bar{x} \\ 0 \\ \bar{z} \end{pmatrix} = R_y(\hat{\alpha}) \begin{pmatrix} \bar{x}' \\ 0 \\ \bar{z}' \end{pmatrix} + \begin{pmatrix} \hat{x} \\ 0 \\ \hat{z} \end{pmatrix}. \quad (2.13)$$

Hence, the optimal transformation moves the centroid of H' onto the centroid of H .

The reason why the authors of [KGP02] restrict the transformation to three DOF is their notion of global motion equivalence, cf. Sect. 2.2. Only admitting rotations about the y axis stems from the following observation: for a wide range of motion classes, including locomotion, it is difficult to perform equivalent motions that only differ with respect to a rotation about any other axis than the y axis. This is due to gravity usually pulling us downwards, in the $-y$ direction. As a counter-example, consider a jumping jack vs. a snow angel, which would be very similar if full six DOF Euclidean motions were admitted. However, the state of “lying on the floor” has strong semantics, and the 3D point cloud distance does well in separating it from “standing upright”.

Only admitting translations in the xz plane boils down to forbidding uneven terrain, which is realistic in many cases. Nevertheless, this restriction may have undesirable consequences in practical applications: sometimes, the ground level of motions changes between recording sessions, leading to a constant difference in the y coordinate—which leads to a large 3D point cloud distance even between otherwise identical motions. Similarly, 3D point cloud distance as defined above is not invariant to scaling. However, such invariance could be incorporated by a simple extension of (2.8).

Just as for the case of quaternion distance (see Sect. 2.5.1), choosing suitable weights w_k , $k \in [1 : K]$, seems to be such a delicate issue that Kovar et al. [KG03, KGP02] simply set $w_k := \frac{1}{K}$ for all k , which amounts to using no weights at all. They note that the weights could be used to stress or mask out certain parts of the body in the comparison, depending on the respective application. The weights could also be used to attribute different importance to different frames within the temporal context $[n - \rho : n + \rho]$, for example to smoothly taper off the influence of the past and the future frames.

2.6 A Semi-Local Similarity Measure

Pullen and Bregler [PB02, Pul02] propose a system for texturing and synthesis of motion capture data based on rough, keyframed motion sketches. For example, an animator might provide a manually keyframed sketch of the lower-body motion for a walking sequence, the objective being to suitably fill in the missing upper-body motion (synthesis) and to apply a realistic look to the lower-body motion (texturing). Both texturing and synthesis require the system to identify mocap clips in a given database that are similar to a manually specified motion query. To measure similarity, Pullen and Bregler start by simultaneously segmenting all angle trajectories of a mocap clip at certain positions in time, which are determined by the local maxima and minima of one of the angle trajectories. To absorb local temporal warpings, the segments are then resampled to unit length and compared using Euclidean distance. More details on their method will be given in Sect. 4.6.

We refer to this way of measuring similarity as *semi-local* because motion-defined local contexts are used for the comparison. The crucial point for such a method to work is that logically similar motions are segmented at logically corresponding points in time. Our technique of temporal segmentation based on relational features, which will be introduced in Sect. 3.2, also yields segments forming logical units. These segments are then used for retrieval, implicitly defining another semi-local similarity measure, see Chap. 4

2.7 Global Similarity Measures

Dynamic time warping (DTW) is a well-known technique to find a global temporal alignment between two given (time-dependent) sequences under certain restrictions. Intuitively, the sequences are warped in a non-linear fashion to match each other as closely as possible. DTW has first been used to compare different speech patterns in automatic speech recognition, see [RJ93], and is similar to the Levenshtein or edit distance [Lev66] for strings. In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data.

To compare two motions D and D' of length N and M frames, respectively, we fix some local distance measure c as introduced above. Evaluating c for all possible pairs of frames taken from D and D' , we obtain an $N \times M$ cost matrix $C = (c(n, m))_{n \in [1:N], m \in [1:M]}$ containing the matching costs for all pairs of frames. For example, Fig. 2.12 (a) shows the cost matrix for the two walking motions of Fig. 2.7 (a) with respect to the quaternion distance, c^{quat} , where fourteen joint rotations (hips, knees, ankles, shoulders, elbows, belly, chest, neck, head) were taken into account. The cost matrix is color-coded: dark colors correspond to low costs, whereas lighter colors correspond to high costs. Contiguous paths within the matrix running through areas of low cost indicate subsequences of the motions that are similar to each other. For instance, the dark path along the main diagonal of the cost matrix in Fig. 2.12 (a) indicates that both motions are globally very similar. Furthermore, the dark path along the secondary diagonal starting at $(160, 1)$ and ending at $(320, 200)$ hints at partial similarity: the first two steps of the straight-line walk are similar to the last two steps of the curved walk. The indices (n, m) of points along such paths encode pairs of frames that can be matched with low cost. DTW exploits these properties of low-cost paths in cost matrices to determine suitable alignments of motions.

A *warping path* is a sequence $w = (w_1, \dots, w_L)$ with $w_\ell = (j_\ell, k_\ell) \in [1 : N] \times [1 : M]$ for $\ell \in [1 : L]$ satisfying the following conditions:

- (i) Boundary condition: $w_1 = (1, 1)$ and $w_L = (N, M)$.
- (ii) Monotonicity condition: $1 \leq j_1 \leq j_2 \leq \dots \leq j_L = N$ and $1 \leq k_1 \leq k_2 \leq \dots \leq k_L = M$.
- (iii) Step size condition: $w_{\ell+1} - w_\ell \in \{(1, 0), (0, 1), (1, 1)\}$.

The total cost of w is defined as

$$c(w) := \sum_{\ell=1}^L c(j_\ell, k_\ell). \quad (2.14)$$

Now, let w^* denote a warping path having minimal total cost among all possible warping paths. Since w^* is in general not uniquely determined, we take the one of smallest length-lexicographical order. Then, the DTW distance $\text{DTW}(D, D')$ between the motions D and D' is defined to be the total cost of w^* ,

$$\text{DTW}(D, D') := c(w^*). \quad (2.15)$$

Note that $\text{DTW}(D, D') = 0$ does not necessarily imply $D = D'$, i. e., DTW is not a metric. An optimal warping path w^* can be computed with $O(NM)$ operations using dynamic programming. For further details on DTW, on the computation of optimal warping paths, as well as extensions and modifications of DTW, we refer to [RJ93].

In the context of motion retrieval, notable applications of DTW and related techniques are the *match web* by Kovar and Gleicher [KG04] as well as the methods by Wu et al. [WCYL03], Liu et al. [LZWM05], or Forbes and Fiume [FF05]. These systems will be reviewed in Sect. 4.6.

2.8 Examples and Comparative Evaluation

We now discuss several examples to illustrate the quaternion and point cloud distance, the concept of cost matrices, and DTW. All example motions have been recorded at a frame rate of 120 Hz. Let us begin by comparing the two walking motions shown in Fig. 2.7 (a) using quaternion distance as well as 3D point cloud distance. We identify the curved walking motion, the trajectories of which are colored red in Fig. 2.7 (a), with $D(n)$ and the straight-line, blue motion with $D'(m)$, where $n \in [1 : 320]$, $m \in [1 : 341]$. The cost matrices¹ for the two motions are shown in Fig. 2.12. In the preceding section, we have already given an interpretation of the quaternion cost matrix C^{quat} shown in Fig. 2.12 (a). The cost matrix C^{3D} for the 3D point cloud distance with $\rho = 0$ as shown in Fig. 2.12 (b) was computed using the same fourteen joints as for the case of the quaternion distance (hips, knees, ankles, shoulders, elbows, belly, chest, neck, head). The two matrices are very similar in overall structure, but C^{3D} appears to be much smoother. Particularly striking are the eight areas of large cost that periodically appear in both cost matrices. These areas arise from the comparison of the most dissimilar poses occurring in a walking cycle: the two extreme poses where the left/right foot is in front of the body while the right/left foot is behind the body. The optimal rotation parameters shown in Fig. 2.12 (e) reveal that the point cloud distance measure tries to align these extreme poses by rotations of up to $\pm 180^\circ$ about the y axis, where the alternating signs of the rotations corresponds to the alternating right/left steps of the gait cycle. Since the left/right extreme poses of normal gait are mirrored copies of each other, they cannot be perfectly aligned by means of a proper rotation, which leads to high costs.

Furthermore, this figure clearly reflects the fact that the red character of Fig. 2.7 (a) is turning to the left by 90° : starting from $n = 1$ and moving upwards along the vertical time axis, one observes an increase of the optimal rotation angle from 0° to 90° in those areas that correspond to the comparison of similar poses from the gait cycles. To interpret the optimal translation parameters shown in Figs. 2.12 (c) and (d), recall that the red character initially stands to the right of the blue character (\rightsquigarrow negative x translation for $n \leq 200$ in Figs. 2.12 (c)), then their paths cross (\rightsquigarrow zero x translation for $n \approx 200$), and finally the red character stands to the left of the blue character (\rightsquigarrow positive x translation for $n > 200$ in Figs. 2.12 (c)). The optimal z translation can be interpreted similarly.

Fig. 2.13 shows the cost matrices for two instances of “rotating both arms forwards”, where the first motion (vertical axis) is much faster than the second one (horizontal axis). As in Fig. 2.8, the first motion is a superposition of walking and rotating the arms. The resulting matrix C^{quat} as shown in Fig. 2.13 (a) seems to be blurred and contains only vague information about a possible alignment. This is due to the vulnerability of the quaternion-based cost measure to angular variations and out-of-phase motions (the left and the right arm are not always perfectly in phase during rotation). As a consequence, the optimal warping path, shown as a cyan-colored line, is jagged and appears to be unreliable due to the lack of clear areas of low cost. By contrast, the matrix C^{3D} clearly exhibits the repetitive structure

¹In the illustrations of cost matrices C , we use a normalized color coding scheme that depends on the minimal and maximal value appearing in C .

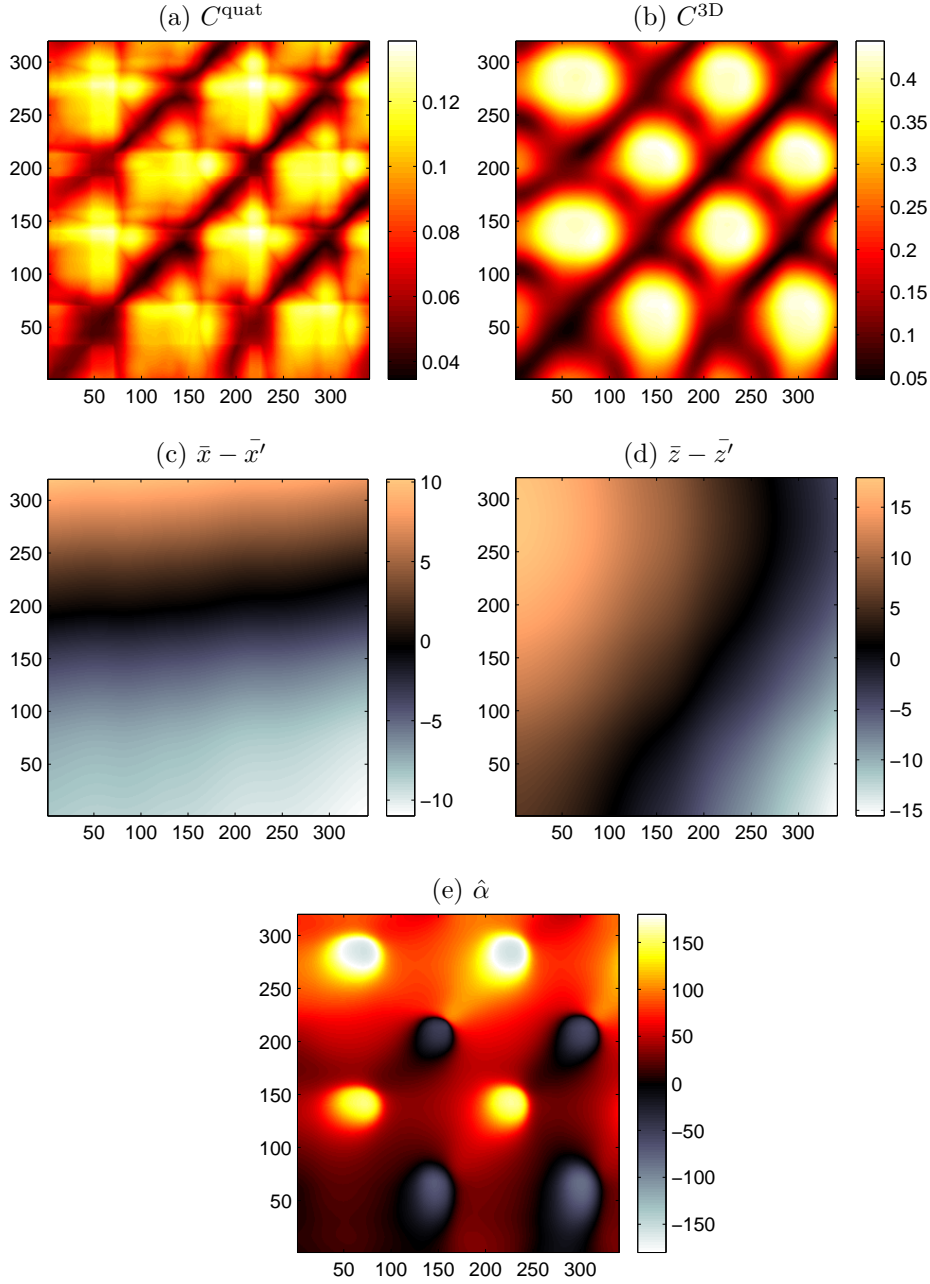


Figure 2.12. (a) Quaternion-based cost matrix C^{quat} for the two walking motions of Fig. 2.7 (a). (b) 3D point cloud-based cost matrix C^{3D} using no temporal context ($\rho = 0$), shown along with the optimal alignment parameters in (c)–(e). The vertical time axes (index n) correspond to the red motion in Fig. 2.7 (a), where the character walks along a 90° left arc. The horizontal time axes (index m) correspond to the blue, straight-line walk. Lengths are expressed in units of 12.54 cm (arbitrary, taken from mocap file format), while the angle parameters $\hat{\alpha}$ are given in degrees. Instead of plotting the parameters \hat{x}_0 and \hat{z}_0 of the respective 2D Euclidean motions, we display in (c) and (d) the x and z translation, respectively, between the point clouds’ centroids. Together with (e), these figures contain the same information but are easier to interpret since they are independent of the rotation.

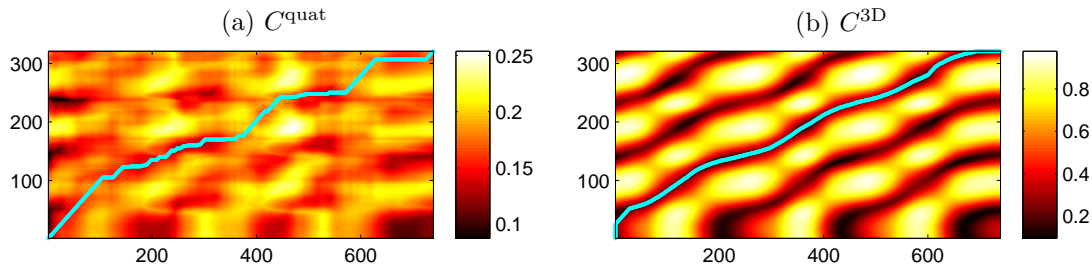


Figure 2.13. Cost matrices for two instances of “rotating both arms forwards”, see Fig. 2.8. The first motion (vertical axis) consists of walking while performing three fast arm rotations, while the actor of the second motion (horizontal axis) is standing while performing three slow arm rotations.

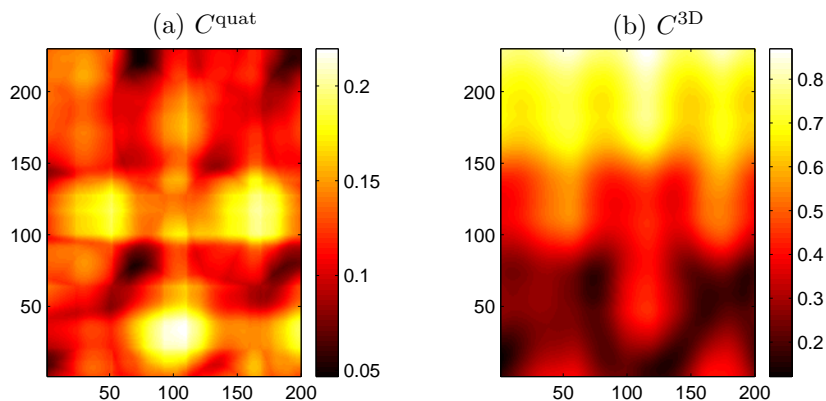


Figure 2.14. (a) Quaternion-based cost matrix C^{quat} for the motions “climbing stairs” and “walking” of Fig. 2.7 (b). (b) 3D point cloud-based cost matrix C^{3D} using no temporal context ($\rho = 0$). The vertical time axes (index n) correspond to the red motion in Fig. 2.7 (b), where the character climbs the stairs. The horizontal time axes (index m) correspond to the blue character.

of the motions since it is less vulnerable to the above mentioned influences. Also note that the slope of the warping path encodes the relative speed difference between the two motions: 100 frames on the vertical axis roughly correspond to 225 frames on the horizontal axis, so the first motion is about 2.25 as fast as the second motion.

In the next example, we compare the motions “climbing a staircase” and “walking at ground level” as shown in Fig. 2.7 (b). Both motions comprise three steps and start with the right foot. The cost matrix with respect to c^{quat} is shown in Fig. 2.14 (a). Note that c^{quat} is invariant under all Euclidean motions of \mathbb{R}^3 including translations in y -direction (vertical direction). Even though the actor in the first motion constantly moves upwards, the overall progression of angle configurations in the climbing motion (first motion) roughly coincides with the one in the regular walking motion (second motion). This is reflected by areas of relatively low costs along the main diagonal. By contrast, the local cost measure c^{3D} is not invariant under translations in y -direction. This leads to high costs when comparing poses towards the end of the first motion (the actor’s body is well above the ground level) with poses of the second motion (the actor’s body is at ground level), see Fig. 2.14 (b).

Our fourth example involves two instances of “lying down on the floor” performed by two different actors. Both motions start from an upright standing pose, and the final poses of the

two motions are shown in Fig. 2.15 (c). The resulting cost matrices with respect to c^{quat} and $c^{3\text{D}}$ are shown in Fig. 2.15 (a) and (b), respectively. Note that the final poses are logically similar (both actors are lying on the floor), while there are large differences in the elbow angles, resulting in significant c^{quat} -costs as shown in the upper right corner (yellow/orange) of Fig. 2.15 (a). This is a typical example of partial (dis-)similarity as discussed above. The difference in the arm’s position has a smaller overall effect on the $c^{3\text{D}}$ -cost, see the upper right corner (black) of Fig. 2.15 (b). A further interesting observation is that the c^{quat} -costs between the standing poses at the beginning of one motion and the lying poses at the end of the other motion are very small (see the area along the line from (500, 30) to (750, 30)). Conversely, the $c^{3\text{D}}$ -costs of the same area are extremely high. This is due to the fact that c^{quat} is invariant under the full rotation group of \mathbb{R}^3 , whereas $c^{3\text{D}}$ is only invariant under rotations about the vertical axis.

Next, we demonstrate the effect of the parameter ρ in the local cost measure $c^{3\text{D}}$, see Sect. 2.5.2. Recall that computing a distance value $c^{3\text{D}}(n, m)$ involves the comparison of entire motion fragments of length $2\rho + 1$ centered at the n^{th} frame of the first motion and at the m^{th} frame of the second motion, respectively. Fig. 2.16 shows the cost matrices for the parameters $\rho = 0$, $\rho = 10$, and $\rho = 20$, where two different jumping jack motions are compared. For the case $\rho = 0$, no temporal context is used, and the value $c^{3\text{D}}(n, m)$ only depends on the n^{th} frame of the first and the m^{th} frame of the second motion. In this case, C low values not only along the main diagonal, but also along the “antidiagonal” (orthogonal to the main diagonal). In other words, the poses of the first jumping jack motion are similar to the corresponding poses of the time-reversed second jumping jack motion, see Fig. 2.16 (a). By increasing ρ , the temporal progression of the frames gain more and more influence, which leads to a smearing effect of the resulting cost matrix along the direction of the main diagonal, see Fig. 2.16 (b) and (c). For example, the value $\rho = 20$ corresponds to a window length of 51 frames, encompassing nearly half a second of motion (at a frame rate of 120 Hz). Recall that we consider the point clouds as *ordered*, so sequence matters during comparison. Hence, in such a temporal context, the poses of the first jumping jack motion are no longer considered to be similar to corresponding poses of the time-reversed second jumping jack (where the poses within the temporal context are not reversed).

As a final example, we investigate the influence of noise on our local cost measures. Fig. 2.17 (a) shows the cost matrix C^{quat} for the comparison of two noisy ballet motions that are similar to the motion shown in Fig. 2.9. The discontinuities in the angle trajectories show as vertical and horizontal stripes of high cost, in certain areas ($n = 100$) heavily occluding the similarity matrix. The influence of the high-frequency noise is not as obvious. By contrast, the $c^{3\text{D}}$ -based distance matrix of Fig. 2.17 (b) does not exhibit any artifacts or noise since the 3D positions of the affected joints can still be matched with relatively low cost.

In general, the 3D point cloud distance is less vulnerable to artifacts than the quaternion-based distance. However, computing cost matrices with respect to $c^{3\text{D}}$ is very time-consuming since optimal alignments have to be computed for every pair of frames. For instance, computing the cost matrix $C^{3\text{D}}$ of Fig. 2.12 (b) took roughly two minutes, while C^{quat} of Fig. 2.12 (a) could be computed in less than two seconds using a Matlab implementation.

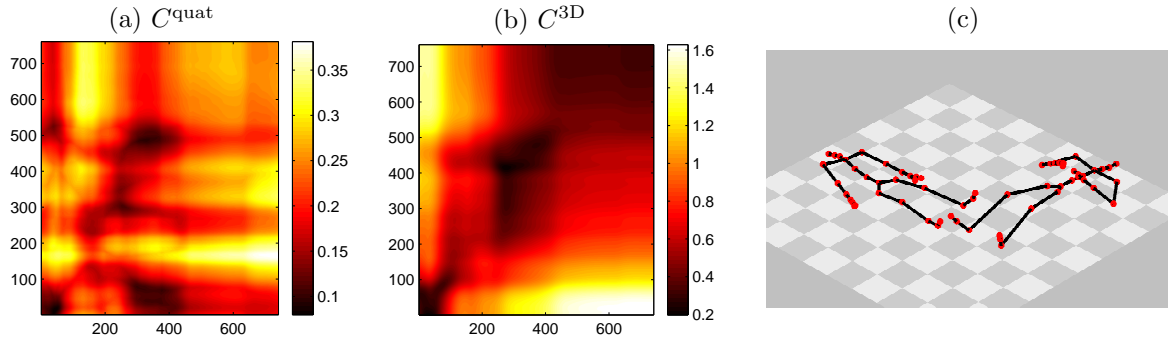


Figure 2.15. (a) Quaternion-based cost matrix C^{quat} for two different actors lying down on the floor. (b) 3D point cloud-based cost matrix C^{3D} using no temporal context ($\rho = 0$). The final poses of the motions are shown in (c).

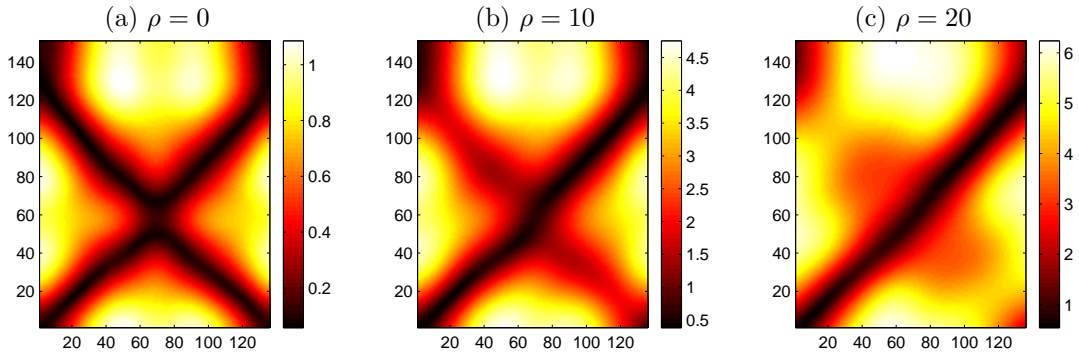


Figure 2.16. Cost matrices corresponding to two jumping jack motions with respect to C^{3D} for different choices of the temporal context parameter, ρ .

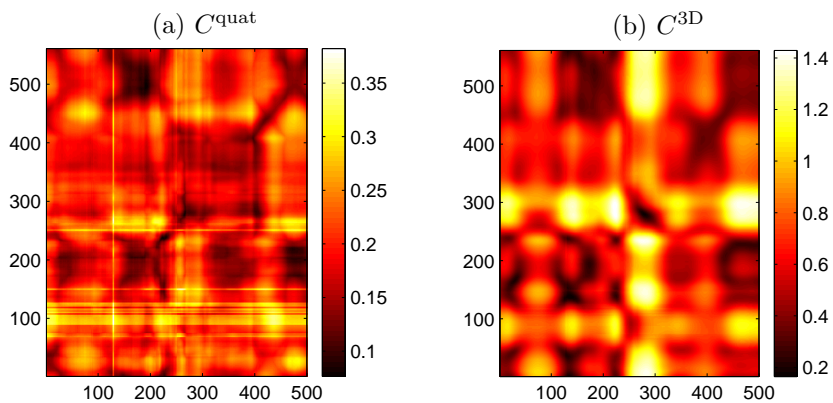


Figure 2.17. Comparison of two different ballet motions that are similar to the motion shown in Fig. 2.9. Both motions contain high-frequency noise as well as discontinuities in some trajectories.

Chapter 3

Relational Motion Features

In a sense, motion capture data has a richer semantic content than, for example, pure video data of a motion, since the 3D position and the meaning of all joints is known for every pose. On the other hand, we have seen that numeric mocap data such as joint rotations or 3D trajectories are not suited for fault-tolerant comparison of motions when focusing on large-scale motion semantics. The main problem is that the granularity of mocap data is too fine for our purpose: irrelevant details, noise, as well as spatial pose deformations may interfere with the actual semantics that we are trying to capture. Furthermore, motions typically exhibit global and local temporal deformations, i. e., different movement speeds and timing differences. Existing approaches to motion comparison handle this problem by means of computationally expensive techniques such as dynamic time warping [KG04] or hidden Markov models [GG04].

To bridge the semantic gap between logical similarity and numerical similarity, we introduce the concept of *relational motion features*, which describe boolean geometric relations between specified points of a pose or between short sequences of poses, see also [MRC05b]. Examples of such features are discussed in Sect. 3.1. Relational features provide a compact, semantically meaningful representation of motions and are largely invariant to various kinds of spatial deformations of poses. Fixing a combination of relational features, we then introduce a feature-driven method for temporal motion segmentation, see Sect. 3.2. It turns out that the sequence of relational features corresponding to the temporal segments is largely invariant to temporal deformations, enabling the use of efficient index-based retrieval techniques.

In view of practical applications, it is important to carefully design relational feature sets so as to capture the aspects of interest while introducing as little redundancy as possible, see Sect. 3.3. Here, we also explain the implementational details of relational features. In Sect. 3.4, we then explicitly state the feature sets that have been used in our experiments on motion retrieval and classification and discuss further examples of relational features applied to concrete motions.

3.1 Examples of Relational Features

In the following discussion, we need the notion of a *boolean feature*, which we define as a function $F : \mathcal{P} \rightarrow \{0, 1\}$ that only assumes the values zero and one. Obviously, any boolean expression of boolean features (evaluated pose-wise) is a boolean feature itself, examples being the conjunction $F_1 \wedge F_2$ and the disjunction $F_1 \vee F_2$ of boolean features F_1 and F_2 .

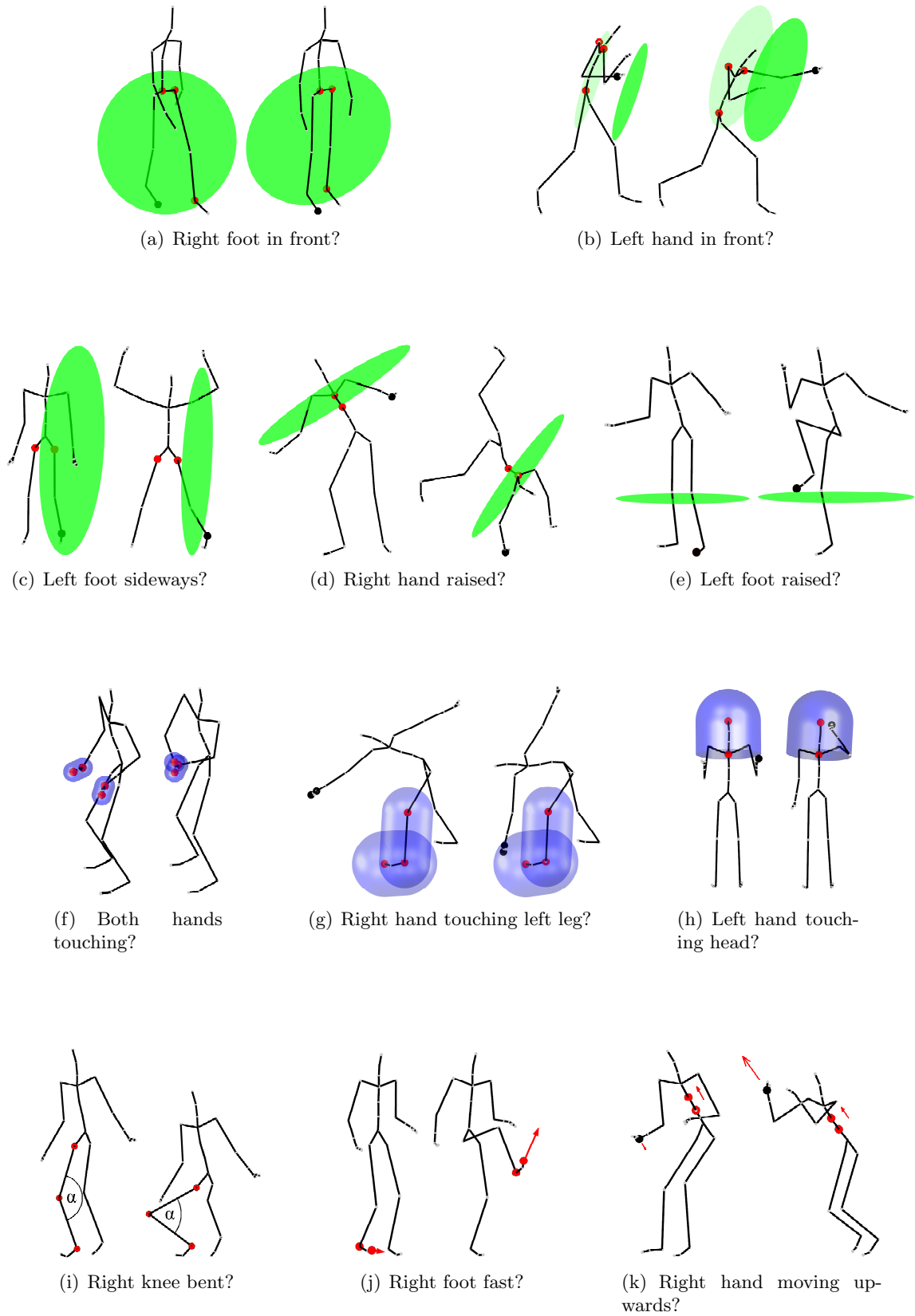


Figure 3.1. Examples of relational features. The joints defining the respective geometric or kinematic entities (planes, velocities, etc.) are highlighted by red markers. If applicable, the joints that are checked against those entities are highlighted by black markers.

Forming a vector of f boolean features for some $f \geq 1$, one obtains a combined feature

$$F : \mathcal{P} \rightarrow \{0, 1\}^f. \quad (3.1)$$

From this point forward, F will be referred to as a *feature function* and the vector $F(P)$ as a *feature vector* or *feature value* of the pose $P \in \mathcal{P}$. Any feature function can be applied to a motion capture data stream $D : [1 : T] \rightarrow \mathcal{P}$ in a pose-wise fashion, which is expressed by the composition $F \circ D$.

Relational features focus on semantically interpretable, boolean aspects of a pose or a short sequence of poses expressing actions or interactions of certain body parts. As a basic example, consider the test whether the right toes lie in front of the plane spanned by the left ankle, the left hip and the root for a fixed pose, cf. Fig. 3.1 (a). More generally, let $p_1, \dots, p_3 \in \mathbb{R}^3$ be four 3D points, the first three of which are in general position. Let $\langle p_1, p_2, p_3 \rangle$ denote the oriented plane spanned by the first three points, where the orientation is determined by point order. We also refer to this plane as the *test plane*. Then define

$$B(p_1, p_2, p_3; p_4) := \begin{cases} 1, & \text{if } p_4 \text{ lies in front of or on } \langle p_1, p_2, p_3 \rangle, \\ 0, & \text{if } p_4 \text{ lies behind } \langle p_1, p_2, p_3 \rangle. \end{cases} \quad (3.2)$$

From this we obtain a *generic relational feature* $F_{\text{plane}}^{(j_1, j_2, j_3; j_4)} : \mathcal{P} \rightarrow \{0, 1\}$ for any four distinct joints $j_i \in J$, $1 \leq i \leq 4$, by defining

$$F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}(P) := B(P^{j_1}, P^{j_2}, P^{j_3}, P^{j_4}). \quad (3.3)$$

The concept of such relational features is simple but powerful, as we will illustrate by continuing the above example. Setting $j_1 := \text{'root'}$, $j_2 := \text{'lankle'}$, $j_3 := \text{'lhip'}$, and $j_4 := \text{'rtoes'}$, we denote the resulting feature by $F^r := F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}$. The test plane determined by j_1 , j_2 , and j_3 is indicated in Fig. 3.1 (a) as a green disc. Since the root joint is placed slightly behind the hip joints in our skeleton, this plane is slightly rotated about the longitudinal axis in a standing pose. Therefore, the feature value $F^r(P)$ is one for a pose P corresponding to a person standing upright and zero when the right foot moves to the back or the left foot to the front, which is typical for locomotion such as walking or running. Interchanging corresponding left and right joints in the definition of F^r and flipping the orientation of the resulting plane, we obtain another feature denoted by F^ℓ , which assumes the value one for a pose P corresponding to a person standing upright and the value zero when the left foot moves to the back or the right foot to the front

Let us have a closer look at the feature function $F := F^r \wedge F^\ell$, which is one if and only if both, the right as well as the left foot, are in front of the respective planes. It turns out that F is very well suited to characterize any kind of locomotion, such as the walking motion $D_{\text{walk}} : [1 : T] \rightarrow \mathcal{P}$ that is shown in Fig. 3.2. The feature sequence $F \circ D : [1 : T] \rightarrow \{0, 1\}$ exhibits exactly two peaks for any locomotion cycle, from which one can easily read off the frequency of the gait cycle (see Fig. 3.3). These peaks correspond to the short phases where the legs pass by each other during walking.

The four joints in $F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}$ can be picked in various meaningful ways. For example, in the case $j_1 = \text{'root'}$, $j_2 = \text{'lshoulder'}$, $j_3 = \text{'rshoulder'}$, and $j_4 = \text{'lwrist'}$, the feature expresses whether the left hand is in front of or behind the body. Introducing a suitable offset, one can alter the semantics of the feature: moving the test plane $\langle P^{j_1}, P^{j_2}, P^{j_3} \rangle$ to the front by one length of the skeleton's humerus, we obtain a feature that can distinguish between a pose

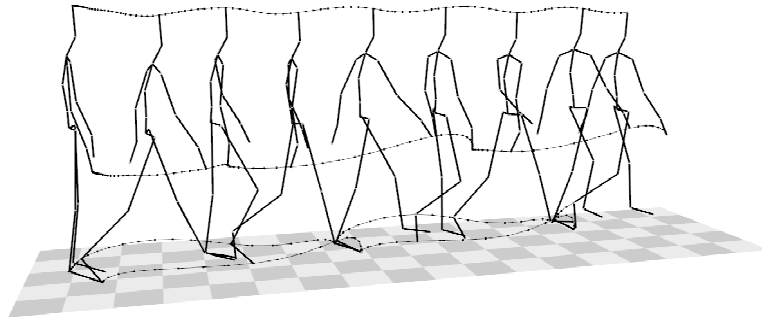


Figure 3.2. A 100-frame walking motion sampled at 30 Hz. The motion starts with a standing pose, then the step sequence left/right/left/right/left is performed. As a reference, the trajectories of the joints ‘headtop’, ‘r ankle’, ‘r toes’, and ‘r fingers’ are shown.

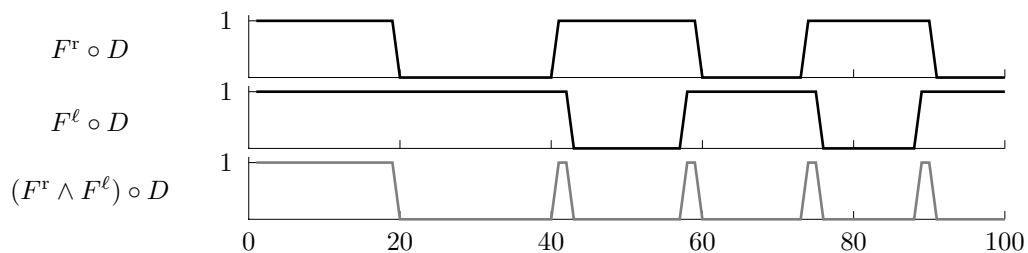


Figure 3.3. Boolean features F^r , F^l , and $F^r \wedge F^l$ applied to the 100-frame walking motion $D = D_{\text{walk}}$ of Fig. 3.2.

with a hand reaching out to the front and a pose with a hand kept close to the body, see Fig. 3.1 (b).

Instead of specifying a test plane by means of three points in general position, it often makes sense to express the plane in terms of a normal vector and a point of attachment. As an example, consider Fig. 3.1 (c), where the test plane is defined to be normal to the line connecting the two hip joints. Tracing along this line one hip width beyond the left hip joint, one reaches the point of attachment for the test plane. By testing the left ankle against this plane, one can then check whether the left foot is stretched out sideways or not.

In Fig. 3.1 (d), the normal of the test plane is given by the vector from the joint ‘chest’ to the joint ‘neck’ and passes through the neck joint. Testing against this plane, one can detect whether a hand is raised above neck height or not. Note that this detection works irrespective of global position, orientation, and scale, since the plane is attached to the skeleton. Considering the second pose of Fig. 3.1 (d), which depicts the middle phase of a cartwheel motion, the condition that the hand is raised is properly detected even though the skeleton is upside-down. This invariance to global position and orientation is a property shared by all relational features that are defined purely in terms of joint-driven geometric entities.

As an example of a relational feature that is not strictly invariant to global orientation, consider Fig. 3.1 (e), where we check whether the left foot is raised or not. Here, the normal of the test plane is defined as the constant “up” vector $(0, 1, 0)^\top$, which does not move along with the skeleton. The point of attachment, however, is defined relative to the skeleton in the following way: locate the joint j that is closest to the ground (minimum y coordinate), taking into account all joints except ‘ltoes’. Then define the point of attachment for the test plane by starting at the 3D position of joint j and moving one length of the humerus in y

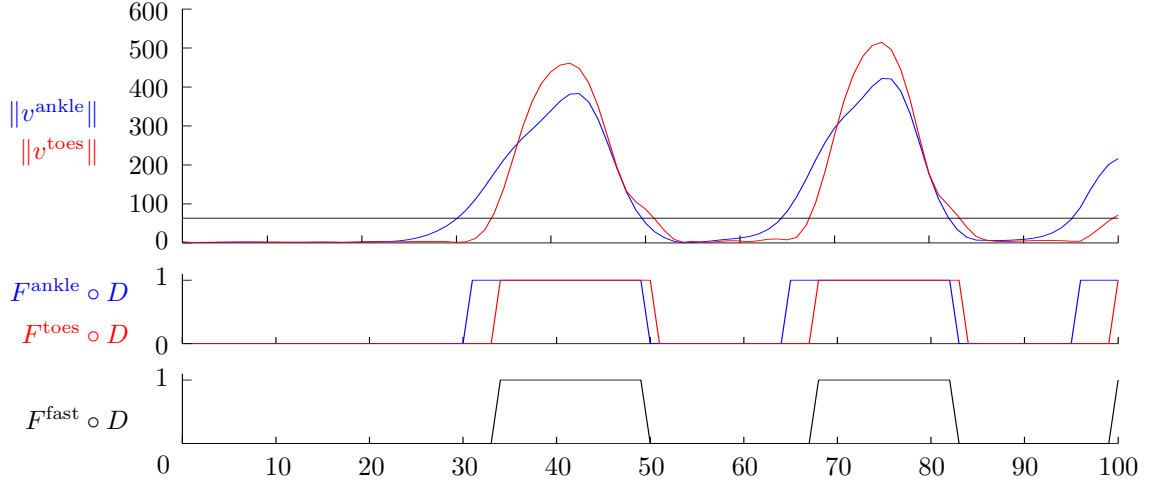


Figure 3.4. Top: Absolute velocities in cm/s of the joints ‘rankle’ ($\|v^{\text{ankle}}\|$, blue) and ‘rtoes’ ($\|v^{\text{toes}}\|$, red) in the walking motion $D = D_{\text{walk}}$ of Fig. 3.2. The black horizontal line at $v_{\text{fast}} = 63$ cm/s indicates the velocity threshold. **Middle:** Thresholded velocity signals for ‘rankle’ and ‘rtoes’. **Bottom:** Feature values for the feature $F^{\text{fast}} = F^{\text{toes}} \wedge F^{\text{ankle}}$ shown in Fig. 3.1 (j). The valley between frames 50 and 67 corresponds to the footplant shown in Fig. 3.5.

direction. Even though the resulting relational feature is not invariant to global rotations of a pose, it grasps the concept of a “raised” foot in all conceivable situations, which is all that matters.

Some instances of a different class of relational features are shown in Fig. 3.1 (f)–(h). Here, we check whether certain parts of the body are close to each other or not. To this end, we consider ε -neighborhoods of certain line segments (often: bones) such as the hands (Fig. 3.1 (f)), the legs and the feet (Fig. 3.1 (g)), or the head (Fig. 3.1 (h)). If a joint lies within such a neighborhood, we assign the feature value one to denote “touching”, otherwise zero to denote “not touching”. Note that ε -neighborhoods of line segments, shown in the figures as blue transparent objects, are composed of two hemispheres joined by a cylinder. In Fig. 3.1 (h), we removed one of the hemispheres since we do not consider the volume below neck height as being close to the head. Of course, the choice of the threshold parameter ε is crucial for the semantics of a relational feature. We will return to the subject of threshold selection in Sect. 3.3.2.

Simply thresholding the joint angles of certain 1 DOF joints by a threshold α can also lead to expressive features, see Fig. 3.1 (i). Here, we check whether the right knee is stretched (feature value zero) or bent (feature value one). Similar features can be defined for the elbow joints. A threshold angle of $\alpha = 120^\circ$ turned out to be well-suited to distinguish the two states “stretched” and “bent”.

So far, our relational features have only referred to a single pose at a time. We now discuss some features that are related to approximated absolute joint velocities, requiring two or more successive poses to compute the “discrete derivative” of joint trajectories. Fig. 3.1 (j) shows the feature $F^{\text{fast}} := F^{\text{toes}} \wedge F^{\text{ankle}}$, which is a movement detector for the right foot. F^{fast} checks whether the absolute velocity of both the right ankle (feature: F^{ankle}) and the right toes (feature: F^{toes}) exceeds a certain velocity threshold, v_{fast} . If so, the feature assumes the value one, otherwise zero, see Fig. 3.4. This feature is well-suited to detect kinematic constraints such as footplants. The reason why we require both the ankle and the toes to be

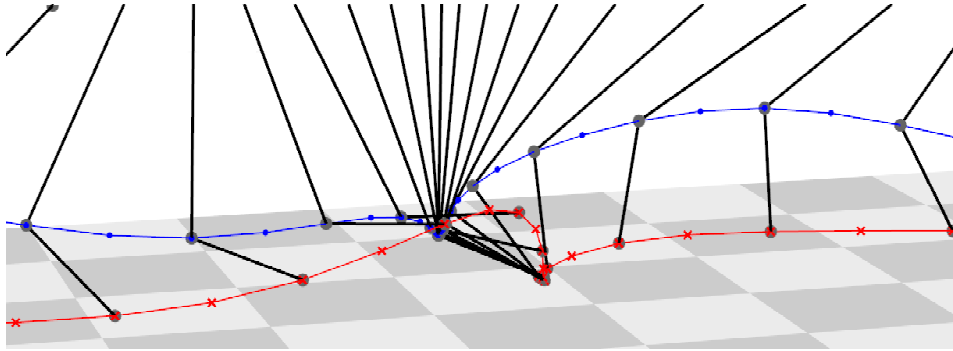


Figure 3.5. Details of the right foot in the walking motion of Fig. 3.2. The trajectories of the joints ‘rankle’ (blue dots) and ‘rtoes’ (red crosses) are shown. Additionally, the right foot and the lower leg have been plotted for every other frame.

sufficiently fast is that we only want to consider the foot as being fast if all parts of the foot are moving. For example, during a typical walking motion, there are phases when the ankle is fast while the heel lifts off the ground, but the toes are still planted on the ground, see also Fig. 3.5. Similarly, there are phases when the heel has already been planted on the ground, leading to a zero velocity for the ankle, while the toes are still rotating downwards. In terms of motion semantics, the foot is not moving in its entirety in both of these situations.

The feature shown in Fig. 3.1 (k) checks whether the right hand is moving into the direction determined by the belly-chest segment. Effectively, this feature detects upward movements of the right hand.

Discussion. In general, feature functions defined purely in terms of geometric or kinematic entities that are expressible by joint coordinates are invariant under global transforms such as Euclidean motions and scalings. Relational features are very coarse in the sense that they express only a single geometric or kinematic aspect, masking out all other aspects of the respective pose. This makes such features robust to variations in the motion capture data stream that are not correlated with the aspect of interest. Using suitable boolean expressions and combinations of several relational features then allows us to focus on or to mask out certain aspects of the respective motion, see Chap. 4.

3.2 Temporal Segmentation

In order to achieve invariance to global and local *temporal* deformations, we now define a feature-dependent temporal segmentation of motion capture data streams. Let $F : \mathcal{P} \rightarrow \{0, 1\}^f$ be a fixed feature function. Then an F -segment of a data stream D is defined to be a substream of D of maximal length consisting of consecutive frames that exhibit the same feature value. Since each segment corresponds to a unique feature vector, the segments induce a sequence of feature vectors, which we also refer to as the F -feature sequence of D and denote by $F[D]$. If M is the number of F -segments of D and if $D(t_m)$ for $t_m \in [1 : T]$, $0 \leq m \leq M$, is any representative pose of the m^{th} segment, then

$$F[D] = (F(D(t_0)), F(D(t_1)), \dots, F(D(t_M))). \quad (3.4)$$

For example, applying the feature function $F^{\text{walk}} := (F^r, F^\ell) : \mathcal{P} \rightarrow \{0, 1\}^2$ to the walking motion $D = D_{\text{walk}}$ of Fig. 3.2 results in a temporal segmentation of D into 10 segments. The corresponding F^{walk} -feature sequence is

$$F^{\text{walk}}[D] = \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right), \quad (3.5)$$

as visualized in Fig. 3.6. A visualization in terms of 3D joint trajectories is given in Fig. 3.7. Obviously, any two adjacent vectors of the sequence $F^{\text{walk}}[D]$ are distinct. The feature sequence clearly exhibits the structure of a walking motion, where phases with parallel feet, reflected by the feature vectors $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, alternate with phases where either the right or the left foot is in front, reflected by the feature vectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, respectively.

Furthermore, changing the combination of features will automatically lead to an adaptation of the induced segmentation. As an example, compare Figs. 3.7 and 3.8. The first figure shows a segmentation with respect to F^{walk} , whereas the second figure shows a segmentation where the second component of F^{walk} has been masked out, leaving only F^r . Note that all segments in Fig. 3.7 are subsegments of the segments in Fig. 3.8, corresponding to the distinction whether the right foot is in front or not. Typically, fine features, i. e., feature functions with many components, induce segmentations with many short segments, whereas coarse features lead to a smaller number of long segments.

The crucial point is that the F -feature sequence of a mocap data stream not only inherits the semantic qualities of the underlying relational features but is also robust to the local and global time variations that typically distinguish logically related motions. The F -feature sequence can thus be viewed as a “fingerprint” of a motion that is largely invariant under spatio-temporal deformations. For example, the walking motion $D = D_{\text{elderlywalk}}$ shown in Fig. 3.9 (a) together with its F^{walk} -segmentation comprises five slow steps performed by an elderly person. Yet, $D_{\text{elderlywalk}}$ has the same F^{walk} -feature sequence as D_{walk} . In general, two motions that differ by some deformation of the time axis will yield the same feature sequences, which is the key property used in our retrieval strategies, cf. Chap. 4.

As a further example, we segment the parallel-leg jumping motion $D = D_{\text{jump}}$ shown in Fig. 3.10 by means of the feature function $F^{\text{jump}} := (F^{\text{rknee}}, F^{\text{lknee}})$, which checks if the right and left knee are bent or stretched. We obtain nine segments corresponding to the feature sequence

$$F^{\text{jump}}[D] = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right). \quad (3.6)$$

The structure of the jumping motion is as follows. Both legs are kept parallel throughout the jump sequence and are stretched during the initial phase of arm-swing (segment 0 in Fig. 3.10). The legs are then bent into a half-squatting position (segments 1–2), preparing the following push-off (starting shortly before segment 3), during which the legs are stretched once more. In the landing phase (starting shortly before segment 5), the legs absorb the energy of the jump by bending as deep as before push-off (segment 6). The jump sequence is concluded by stretching the legs into a normal standing position (segments 7–8). Such a sequence of bending and stretching the knees is characteristic for many kinds of parallel-leg jumping motions. However, segments number 1, 3, 5, and 7 are *transitory segments* of very short duration. They assume one of the feature values $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ or $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ more or less randomly, depending on which of the knees is bent/stretched first during the push off and landing phases. We will explain in Chap. 4 how to manually exclude transitory segments in the comparison of motions to some extent. In Chap. 5, we will then introduce an automatic way of detecting and excluding such segments.

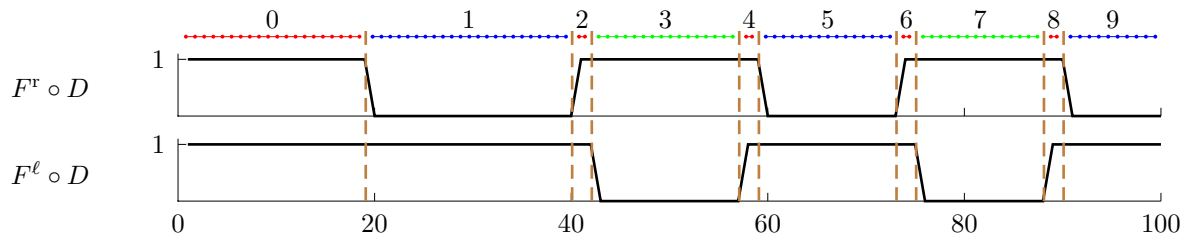


Figure 3.6. Boolean features F^r and F^ℓ applied to the 100-frame walking motion $D = D_{\text{walk}}$ of Fig. 3.2. The temporal segmentation induced by the combined feature function $F^{\text{walk}} = (F^r, F^\ell)$ is indicated by vertical lines as well as colored bars and segment numbers corresponding to Fig. 3.7.

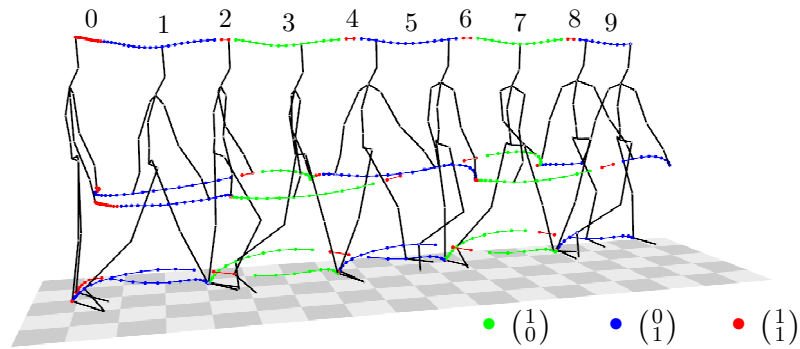


Figure 3.7. F^{walk} -segmentation of $D = D_{\text{walk}}$. F^{walk} -equivalent poses are indicated by identically colored trajectory segments. The trajectories of the joints ‘headtop’, ‘rankle’, and ‘rfingers’ are shown.

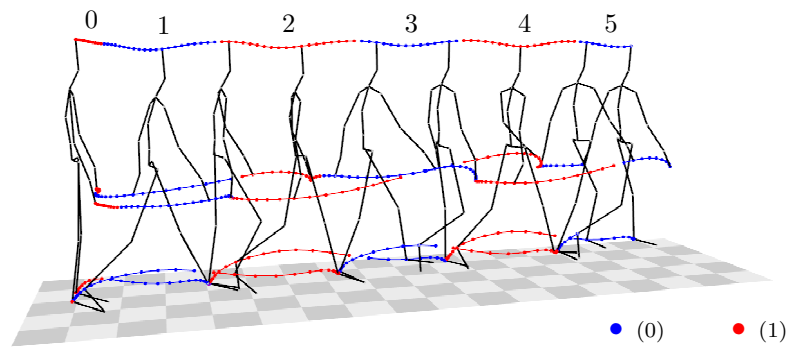


Figure 3.8. F^r -segmentation of $D = D_{\text{walk}}$.

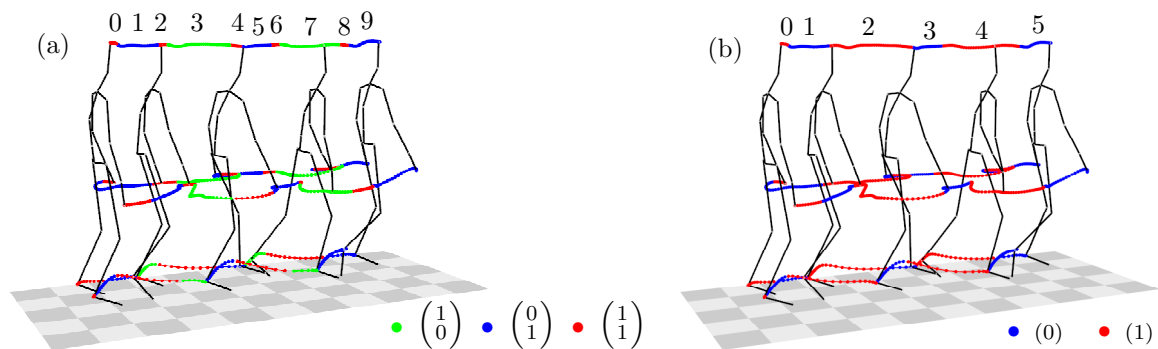


Figure 3.9. (a) F^{walk} -segmentation and (b) F^r -segmentation of $D = D_{\text{elderlywalk}}$.

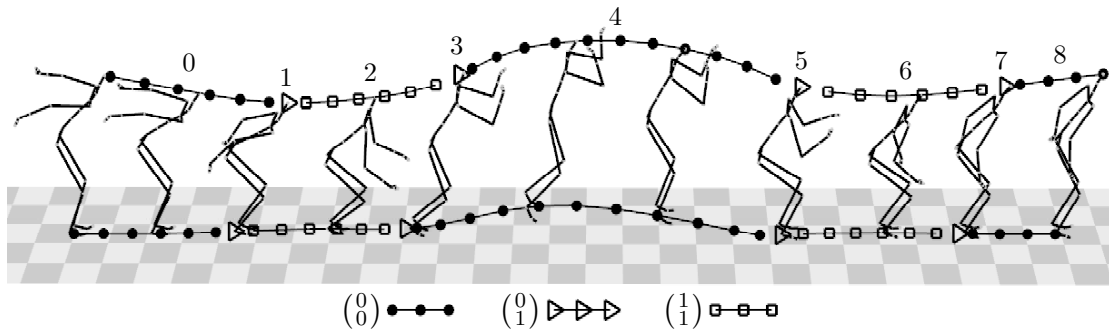


Figure 3.10. F^{jump} -segmentation of a parallel leg jumping motion $D = D_{\text{jump}}$, where F^{jump} -equivalent poses are indicated by identically marked trajectory segments. The trajectories of the joints ‘headtop’ and ‘rankle’ are shown.

As an example involving a more complex motion and a more complex feature function, we consider a kick/punch combination, see Fig. 3.11. Here, we use a feature function $F^4 : \mathcal{P} \rightarrow \{0, 1\}^4$ consisting of the following four boolean features: “right knee bent”, (feature F_{20} of Table 3.2), “left elbow bent” (F_8), “right foot raised” (F_{17}), and “left hand reaching out to the front” (feature E_2 of Table 3.1). Note that there are 16 different feature combinations with respect to F^4 . Fig. 3.11 shows the 11 segments of the resulting F^4 -segmentation.

Discussion. The main idea is that two motion capture data streams D_1 and D_2 can now be compared via their F -feature sequences $F[D_1]$ and $F[D_2]$ instead of comparing the data streams on a frame-to-frame basis. This has several advantages:

1. Since spatial and temporal invariance are already incorporated in the features and segments, one can use efficient string matching methods to compare the data streams instead of applying cost-intensive techniques such as DTW at the frame level.
2. One can decide which aspects of the motions to focus on by picking a suitable feature function F .
3. In general, the number of segments is much smaller than the number of frames, which accounts for efficient computations.

However, we have also seen that aspects of a motion that change nearly simultaneously can lead to short transitory segments with unpredictable feature values, necessitating some kind of fault tolerance mechanism for the comparison of feature sequences.

3.3 Feature Design

To facilitate retrieval in large motion databases containing a great variety of motions, it is essential to provide the end-user of our retrieval system with a semantically rich set of features covering different body parts and various aspects of motions. Of course, the requirements to such a feature set will heavily depend on the intended application. The goal of our retrieval system is to search for motion clips in view of their rough course of motion.

In this section, we describe our approach to designing such feature sets. Most features have been derived from *generic* relational features, see Sect. 3.3.1, which depend on a set of joints and some threshold values. Here, an appropriate selection of thresholds is crucial to

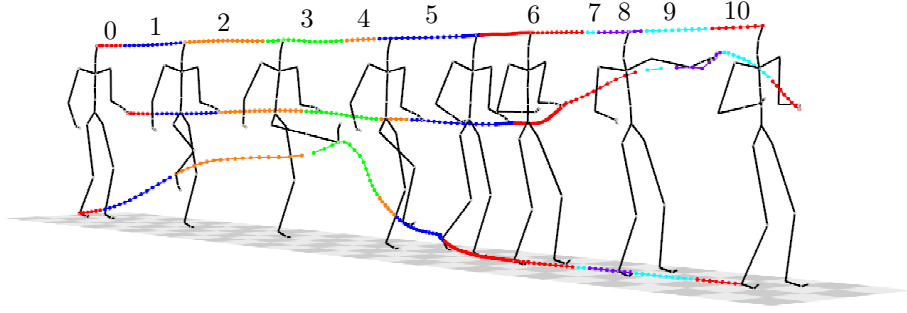


Figure 3.11. Right foot kick followed by a left hand punch. The trajectories of the joints ‘headtop’, ‘rankle’, and ‘lfingers’ are shown. The segments are induced by a 4-component feature function F^4 comprising the thresholded angles for ‘rknee’ and ‘lelbow’ as well as features describing the relations “right foot raised” and “left hand in front”.

designing semantically meaningful features. This topic will be discussed in Sect. 3.3.2. We have also tested automatic methods of selecting appropriate joint combinations to be used with generic features, see Sect. 3.3.4.

3.3.1 Generic Relational Features

Most of our features have been constructed from the following seven generic relational features, which encode certain geometric and kinematic joint constellations. Recall from Sect. 3.1 the discussion of the generic feature F_{plane} , which checks whether a certain joint lies in front of a plane defined by three other joints. Similarly, each of the generic features

$$\begin{aligned}
 F_{\text{plane}} &= F_{\theta, \text{plane}}^{(j_1, j_2, j_3; j_4)} \\
 F_{\text{nplane}} &= F_{\theta, \text{nplane}}^{(j_1, j_2, j_3; j_4)} \\
 F_{\text{touch}} &= F_{\theta, \text{touch}}^{(j_1, j_2; j_3)} \\
 F_{\text{angle}} &= F_{\theta, \text{angle}}^{(j_1, j_2; j_3, j_4)} \\
 F_{\text{fast}} &= F_{\theta, \text{fast}}^{(j_1)} \\
 F_{\text{move}} &= F_{\theta, \text{move}}^{(j_1, j_2, j_3; j_4)} \\
 F_{\text{nmove}} &= F_{\theta, \text{nmove}}^{(j_1, j_2, j_3; j_4)}
 \end{aligned} \tag{3.7}$$

maps a given pose to the set $\{0, 1\}$ and depends on a number of joints, denoted by j_1, \dots, j_4 . The term *relational feature* hails from the fact that a generic feature depending on n joints can be viewed as an n -ary relation on the set J , given a fixed pose.

The latter three features involve velocities and therefore assume a short sequence of poses centered at the current pose to be given. In contrast to the notation introduced in Sect. 3.1, all generic features now depend on a threshold value or threshold range θ . All of our features can be computed in linear time, where the complexity parameter is the length T of the respective mocap data stream.

The generic feature $F_{\theta, \text{plane}}^{(j_1, j_2, j_3; j_4)}$

Given four points $p_1, \dots, p_4 \in \mathbb{R}^3$, the signed distance of p_4 to the plane $\langle p_1, p_2, p_3 \rangle$ can be computed as follows. First, determine a unit normal vector

$$n(p_1, p_2, p_3) := \frac{(p_1 - p_3) \times (p_2 - p_3)}{\|(p_1 - p_3) \times (p_2 - p_3)\|}, \quad (3.8)$$

the orientation of which is given by the right-hand-rule. Then evaluate the Hesse normal form, which yields the signed distance of p_4 as

$$d_{\text{plane}}(p_1, p_2, p_3; p_4) := \langle n(p_1, p_2, p_3), p_4 - p_3 \rangle, \quad (3.9)$$

where positive values of $d(p_1, p_2, p_3; p_4)$ indicate that p_4 lies in the open half space into which the normal $n(p_1, p_2, p_3)$ points. Next, we apply the thresholding function

$$H_{\theta}(x) := \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{otherwise,} \end{cases} \quad (3.10)$$

where $\theta, x \in \mathbb{R}$. For a given pose $P \in \mathcal{P}$, we can then compute the feature value as

$$F_{\theta, \text{plane}}^{(j_1, j_2, j_3; j_4)}(P) = H_{\theta}(d_{\text{plane}}(P^{j_1}, P^{j_2}, P^{j_3}; P^{j_4})). \quad (3.11)$$

In other words, F_{plane} assumes the value one iff joint j_4 has a signed distance of at least θ from the oriented plane spanned by the joints j_1, j_2 and j_3 . Compare the examples in Figs. 3.1 (a), (b).

The generic feature $F_{\theta, \text{nplane}}^{(j_1, j_2, j_3; j_4)}$

A similar test is performed by $F_{\theta, \text{nplane}}^{(j_1, j_2, j_3; j_4)}$, but here we define the plane directly in terms of a normal vector and an anchor point. For $p_1, \dots, p_4 \in \mathbb{R}^3$, let

$$n(p_1, p_2) := \frac{p_2 - p_1}{\|p_2 - p_1\|}. \quad (3.12)$$

Then the Hesse normal form yields the signed distance of p_4 as

$$d_{\text{nplane}}(p_1, p_2, p_3; p_4) := \langle n(p_1, p_2), p_4 - p_3 \rangle, \quad (3.13)$$

where the plane passes through the anchor point p_3 . This allows us to define the feature value for a given pose $P \in \mathcal{P}$ as

$$F_{\theta, \text{nplane}}^{(j_1, j_2, j_3; j_4)}(P) := H_{\theta}(d_{\text{nplane}}(P^{j_1}, P^{j_2}, P^{j_3}; P^{j_4})). \quad (3.14)$$

Figs. 3.1 (c), (d) demonstrate the use of this generic feature.

The generic feature $F_{\theta, \text{touch}}^{(j_1, j_2; j_3)}$

Given three points $p_1, p_2, p_3 \in \mathbb{R}^3$, this feature returns one iff p_3 lies within the θ -neighborhood of the line segment L connecting p_1 and p_2 . The surface of this θ -neighborhood is a cylinder with radius θ and axis L that is capped by hemispheres of radius θ at both ends, see Fig. 3.1 (f),(g) for an illustration. A unit normal vector for the bottom and top planes of the cylinder points in the direction of the cylinder axis (directed from p_1 to p_2) is given by $n(p_1, p_2)$, cf. (3.12). The distance of p_3 to the line $\langle p_1, p_2 \rangle$ can then be computed as

$$d_{\text{line}}(p_1, p_2; p_3) := \|(p_3 - p_1) \times n(p_1, p_2)\| = |\sin \alpha| \|p_3 - p_1\|, \quad (3.15)$$

where α is the angle enclosed between $p_3 - p_1$ and n . Since we do not want to consider the entire line $\langle p_1, p_2 \rangle$ but rather the line segment L , the distance function d_{line} is only valid as long as p_3 is closer to an inner point of L than to one of its endpoints. In the latter case, we must use the distance functions

$$d_{\text{point}}(p_1; p_3) := \|p_3 - p_1\| \quad \text{and} \quad d_{\text{point}}(p_2; p_3) := \|p_3 - p_2\| \quad (3.16)$$

To find out which of the distance functions applies, we use the distance function for the bottom and top planes of the cylinder, which are

$$d_1(p_1, p_2; p_3) := \langle n(p_1, p_2), p_3 - p_1 \rangle \quad \text{and} \quad d_2(p_1, p_2; p_3) := \langle n(p_1, p_2), p_3 - p_2 \rangle, \quad (3.17)$$

respectively. This yields the distance function

$$d_{\text{segment}}(p_1, p_2; p_3) := \begin{cases} d_{\text{point}}(p_1; p_3) & \text{if } d_1(p_1, p_2; p_3) \leq 0 \\ d_{\text{line}}(p_1; p_3) & \text{if } d_1(p_1, p_2; p_3) > 0 \wedge d_2(p_1, p_2; p_3) < 0 \\ d_{\text{point}}(p_2; p_3) & \text{if } d_2(p_1, p_2; p_3) \geq 0. \end{cases} \quad (3.18)$$

The resulting feature value for a pose $P \in \mathcal{P}$ is

$$F_{\theta, \text{touch}}^{(j_1, j_2; j_3)}(P) := 1 - H_{\theta}(d_{\text{segment}}(P^{j_1}, P^{j_2}; P^{j_3})), \quad (3.19)$$

where the subtraction from 1 amounts to a logical negation, which is necessary since we want to assign a feature value of one if the distance is smaller than θ . In order to obtain a decision surface where the lower hemisphere is removed (as shown in Fig. 3.1 (h)), one can alter the distance function in the following way:

$$d'_{\text{segment}}(p_1, p_2; p_3) := \begin{cases} \infty & \text{if } d_1(p_1, p_2; p_3) \leq 0 \\ d_{\text{line}}(p_1; p_3) & \text{if } d_1(p_1, p_2; p_3) > 0 \wedge d_2(p_1, p_2; p_3) < 0 \\ d_{\text{point}}(p_2; p_3) & \text{if } d_2(p_1, p_2; p_3) \geq 0. \end{cases} \quad (3.20)$$

Analogous modifications allow for the removal of the upper hemisphere or both of the hemispheres. Also note that this feature can check whether two joints are close to each other by setting $j_1 = j_2$.

The generic feature $F_{\theta, \text{angle}}^{(j_1, j_2; j_3, j_4)}$

This feature assumes the value one iff the angle enclosed between the directed segments determined by (j_2, j_1) and (j_3, j_4) is within the angle range $\theta \subseteq [0, \pi]$. Given four points

$p_1, \dots, p_4 \in \mathbb{R}^3$, we start by computing the unit vectors $n(p_1, p_2)$ and $n(p_3, p_4)$ pointing in the directions of the two segments, cf. (3.12). The angle enclosed between $n(p_1, p_2)$ and $n(p_3, p_4)$ is denoted as $\alpha(p_1, p_2; p_3, p_4) \in [0, \pi)$; more precisely, we mean the smaller of the enclosed angles. This angle can now be computed as

$$\alpha(p_1, p_2; p_3, p_4) := \arccos\langle n(p_1, p_2), n(p_3, p_4) \rangle. \quad (3.21)$$

Note that $n(p_1, p_2)$ points from p_1 to p_2 and $n(p_3, p_4)$ points from p_3 to p_4 . In case $p_1 = p_3 = p$, this definition allows for the interpretation that there is a common joint between the two segments joining at p , both of which point away from p . The feature value for a pose $P \in \mathcal{P}$ is defined as

$$F_{\theta, \text{angle}}^{(j_1, j_2; j_3, j_4)}(P) := \chi_\theta(\alpha(P^{j_1}, P^{j_2}; P^{j_3}, P^{j_4})), \quad (3.22)$$

where $\chi_\theta : [0, \pi] \rightarrow \{0, 1\}$ is the characteristic function of the angle range $\theta \subseteq [0, \pi]$. See also Fig. 3.1 (i).

There are three reasons why we recompute angles from 3D joint positions instead of directly using the joint angle information from the animated skeleton. First, we aim at keeping our features independent of specific data formats, focusing exclusively on 3D joint coordinates. Second, the joint angles that are specified in typical mocap files are usually offset against our features' joint angles by an angle that may vary between mocap files; for example, the knee angle for a stretched knee is denoted as 0° in many mocap files, whereas the above computations would yield 180° . Finally, our method also works for two bones or two joint-driven line segments that are not connected by a common joint; for example, it could make sense to measure the angle of the arm against the direction of the spine.

The generic feature $F_{\theta, \text{fast}}^{(j_1)}$

The three remaining generic features operate on velocity data that is approximated from the 3D joint trajectories of the input motion. Given a mocap data stream $D : [1 : T] \rightarrow \mathcal{P}$, we focus on a single joint $j \in \mathcal{J}$ and consider its 3D trajectory D^j , which we think of as a sequence $(p(1), \dots, p(T))$ in \mathbb{R}^3 . From this trajectory, we can compute approximate velocity vectors by taking the “discrete derivative”

$$v(t) := \begin{cases} \frac{p(t+1) - p(t)}{\Delta t} & \text{if } 1 \leq t < T \\ v(T-1) & \text{if } t = T, \end{cases} \quad (3.23)$$

where Δt is the inverse of the sampling rate, for example $\Delta t = \frac{1}{120}$ s in our ballet motion. Note that we duplicate the last velocity vector to maintain the length of the original trajectory.

Mocap data may contain significant high-frequency noise components, which typically leads to very noisy velocity data since the discrete derivative (3.23) corresponds to a highpass filter. For example, consider the toe trajectory shown in the ballet motion of Fig. 2.9, where the noise shows as extremely irregular sample spacing. The trajectory even contains some obvious outliers on the left hand side of the figure. The black curve in Fig. 3.12 corresponds to the magnitude of the 3D velocity vectors of the noisy toe trajectory and clearly shows some major peaks corresponding to the outliers as well as high-frequency noise at all times.

Such spatial noise can be due to calibration and tracking errors, to inadequate data cleanup such as unresolved marker occlusions and marker confusions, and to skeletal fitting errors. We make the simple assumption that the three dimensions of the noise components in the

3D velocity vectors are uncorrelated and the underlying stochastic process is stationary, so suitable averaging over several velocity vectors should allow for some of the noise to cancel out. To this end, we apply a moving average filter to the 3D velocity vectors.

We start by extending the sequence of velocity vectors using *symmetric padding* before frame 1 and after frame T , yielding the infinite sequence $\tilde{v} : \mathbb{Z} \rightarrow \mathbb{R}^3$ defined by

$$\tilde{v}(t) := \begin{cases} \tilde{v}(1-t) & \text{if } t < 1 \\ \tilde{v}(2T+1-t) & \text{if } t > T \\ v(t) & \text{otherwise.} \end{cases} \quad (3.24)$$

The convolution filter C^h convolves its input sequence with the filter coefficients $(h_k)_{k \in [-K:K]}$, where convolution between a one-dimensional sequence and a sequence of 3D vectors is meant in a coordinate-wise sense. Choosing

$$h_k := \frac{e^{-\frac{1}{2}(\beta \frac{2k}{2K+1})^2}}{\sum_{k=-K}^K e^{-\frac{1}{2}(\beta \frac{2k}{2K+1})^2}} \quad (3.25)$$

for $k \in [-K : K]$, we obtain a *Gaussian lowpass filter mask* of length $2K + 1$, where the parameter β corresponds to the inverse of the variance of the corresponding Gaussian distribution. We chose $\beta = 2.5$, see also [Har78]. The noise-reduced sequence is then given by

$$\bar{v}(t) := C^h[\tilde{v}](t) = (h * \tilde{v})(t) = \sum_{k=-K}^K h_k \tilde{v}(t-k), \quad (3.26)$$

which we evaluate for $t \in [1 : T]$. Note that this is equivalent to filtering the 3D data prior to computing the discrete derivative due to the linearity of convolution.

The red curve in Fig. 3.12 depicts the magnitude of the 3D velocity after filtering with the Gaussian lowpass filter of length $2K+1 = 13$, which roughly corresponds to the experimentally determined window length of 100 ms. Most of the noise has been removed. This example also demonstrates the noise reduction that is due to the averaging of uncorrelated 3D noise components, which tend to cancel out before the absolute value is computed: the spikes around frames 80, 125, and 240 would lead to much larger responses if the absolute value had been applied prior to the filtering step. Of course, our filtering approach to noise removal may as well remove some semantic details, but this is irrelevant since the focus of our features is not on such details but rather on coarse, global properties. A different, quaternion-based filtering approach is discussed in App. B.

Writing the filtered velocity sequence corresponding to the trajectory of joint $j \in J$ in operator notation as $\bar{v}[D^j]$, we then define the feature value for a pose $D(t) = P \in \mathcal{P}$ as

$$F_{\theta, \text{fast}}^{(j_1)}(D(t)) := H_{\theta}(\|\bar{v}[D^{j_1}](t)\|). \quad (3.27)$$

Note that we require P to be embedded within the context of a mocap data stream D .

The generic feature $F_{\theta, \text{nmove}}^{(j_1, j_2, j_3; j_4)}$

This feature considers the velocity of joint j_4 relative to joint j_3 and assumes the value one iff the component of this velocity in the direction determined by the line segment from j_1 to j_2 is above θ . The said velocity component can also be viewed as the signed, one-dimensional

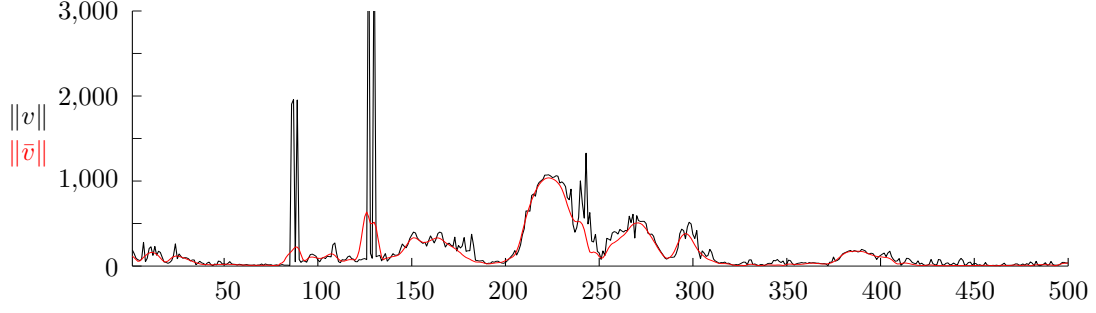


Figure 3.12. Absolute velocity in cm/s of the joint ‘toes’ in the ballet motion of Fig. 2.9. Two velocity curves have been computed: the black curve shows the magnitude of the 3D velocity vectors resulting from raw 3D data, whereas the red curve shows the corresponding values after the 3D velocity vectors have been filtered with a Gaussian lowpass filter of length 108 ms, or 13 frames.

velocity of joint j_4 relative to the plane determined by a normal vector (given by j_1 and j_2) and the anchor joint j_3 . This was the motivation for the abbreviation “nmove”, where the ‘n’ stands for “normal”. See Fig. 3.1 (k) for an example.

We compute the desired velocity component in the following way. Given four points $p_1 = D^{j_1}(t), \dots, p_4 = D^{j_4}(t) \in \mathbb{R}^3$ from frame $t \in [1 : T]$ of a mocap data stream D , we first determine the unit normal vector $n(p_1, p_2)$, cf. (3.12). The filtered relative velocity vector

$$\bar{v}(p_3, p_4) := \bar{v}[D^{j_4} - D^{j_3}](t) \quad (3.28)$$

between joints j_3 and j_4 at time t is then projected onto $n(p_1, p_2)$ by means of the inner product, yielding the scalar velocity

$$\bar{v}(p_1, p_2, p_3; p_4) := \langle n(p_1, p_2), \bar{v}(p_3, p_4) \rangle, \quad (3.29)$$

similar to (3.9). The feature value for a pose $D(t) = P \in \mathcal{P}$ is then defined as

$$F_{\theta, \text{nmove}}^{(j_1, j_2, j_3; j_4)}(D(t)) := H_{\theta}(\bar{v}(D^{j_1}(t), D^{j_2}(t), D^{j_3}(t); D^{j_4}(t))) \quad (3.30)$$

for a suitable velocity threshold θ .

Fig. 3.13 shows the output of an upward movement detector for the right hand, as introduced in Fig. 3.1 (k), applied to a jumping motion. There is a clear peak in the velocity curve corresponding to the upward motion of the hand during the phase of building up momentum for the jump. Also note the rotational invariance of the feature: during the jumping motion shown in Fig. 3.10, the upper body leans forwards. The “up” vector, which is identified with the direction of the spine, follows this forward rotation such that the feature still returns the value one during the arm swing.

The generic feature $F_{\theta, \text{move}}^{(j_1, j_2, j_3; j_4)}$

This feature has similar semantics as $F_{\theta, \text{nmove}}^{(j_1, j_2, j_3; j_4)}$, but the direction is given by the normal vector of the oriented plane spanned by j_1 , j_2 , and j_3 .

Other generic features

We also considered the absolute value of the relative velocity between two points p_1 and p_2 , $\|\bar{v}(p_1, p_2)\|$, cf. (3.28). This can be useful to eliminate the superposition of certain movements.

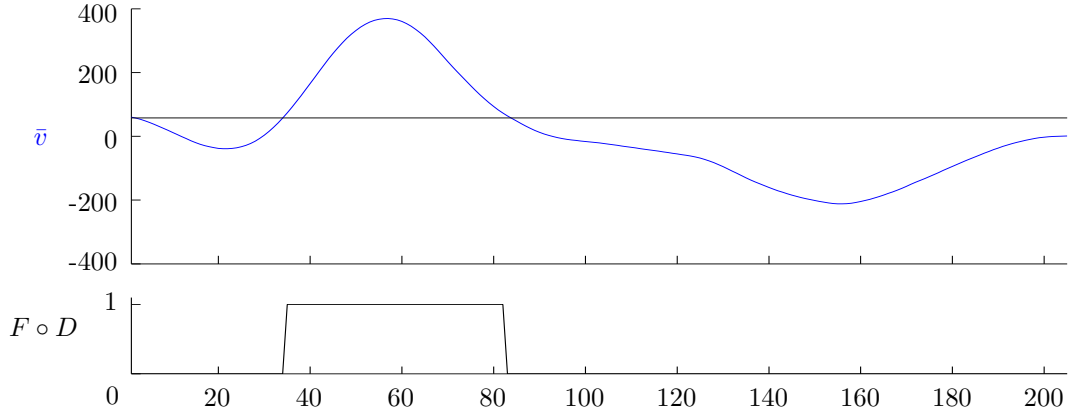


Figure 3.13. Top: Velocity in cm/s of the joint ‘rwrst’ relative to the ‘belly’–‘chest’ direction in the jumping motion $D = D_{\text{jump}}$ of Fig. 3.10. The black horizontal line at $v_{\text{fast}} = 58$ cm/s indicates the velocity threshold. **Bottom:** Feature values for $F = F_{58, \text{nmov}}^{(\text{‘belly’}, \text{‘chest’}, \text{‘chest’}, \text{‘rwrst’})}$ on D .

For example, during walking or running motions, the observed velocity of the foot can be decomposed into the sum of the root’s forward velocity and the velocity of the foot relative to the root. We eliminated the root’s forward velocity by modifying the feature shown in Fig. 3.1 (j) so as to check the velocity of the right foot relative to the root. However, this has the major disadvantage that the action of standing on a fixed spot while moving the hips leads to a nonzero velocity of the foot, which renders such a feature inappropriate for detecting floor contacts. But this was one of the main intentions behind the velocity features for the feet. In general, relative velocities with respect to certain joints often lead to unforeseen effects that preclude a meaningful comparison of motions. For these reasons, this type of features has been dropped.

For the future, it would be interesting to investigate how dynamic parameters of a motion (joint torques, forces, center-of-mass trajectories) can be used in the context of relational features. In applications of both motion synthesis and analysis [ALP04, LHP05, NF05], it is pointed out that dynamic parameters encode important information about motion style. On the other hand, dynamic parameters can also provide important clues about the motion content, see also [Krü06]. For example, walking in a slight curve to the left/right might be distinguishable by certain asymmetries in the legs’ joint torques, whereas purely kinematic parameters such as joint trajectories or velocities would not allow for such a distinction. On the downside, dynamic parameters such as joint torques usually have to be derived from the raw mocap data by means of inverse dynamics [KB96], which is highly sensitive to noise and involves heuristics for resolving the weight distribution between the two legs.

3.3.2 Choosing Thresholds

Selecting appropriate generic features and suitable combinations of joints only determines a part of the semantics of a relational feature. The other part comes from a semantically meaningful choice of the threshold parameter θ . As an important point, note that variations of θ may completely change the meaning of a relational feature, which in turn influences the motion aspect that can be grasped by that feature. According to the different types of generic features, there are different interpretations of such threshold values θ :

- Plane offsets $\theta \in \mathbb{R}$ for the generic features $F_{\theta, \text{plane}}$ and $F_{\theta, \text{nplane}}$ (see Fig. 3.1 (b),(c),(e)),

can be thought of as shifting the respective plane in the direction of the plane’s normal vector. For example, choosing $\theta = 0$ for the feature shown in Fig. 3.1 (b) yields a detector for whether the hand is in front of the body or not. Increasing θ , one obtains detectors for different magnitudes of reaching out in front of the body.

- Proximity thresholds $\theta \in \mathbb{R}$ for the generic feature $F_{\theta, \text{touch}}$ (see Fig. 3.1 (f)–(h)) define the notion of closeness between a joint and a body segment. As an example, one could significantly increase θ for the “hands touching” detector shown in Fig. 3.1 (f), say, to three upper arm’s lengths. The resulting feature would output the value one for most realistic poses, except when the hands are deliberately stretched far apart from each other. One could then use the negation of such a feature to detect “hands far apart”.
- Angle ranges $\theta \subseteq [0, \pi]$ for the generic feature $F_{\theta, \text{angle}}$ may be used to define the notion of a “bent” or “stretched” joint, or they can define more general angle relations between joint-driven or absolute segments such as the “spine horizontal” detector (feature F_{32} of Tab. 3.2). The detector for the condition “knee bent” shown in Fig. 3.1 (i)) uses the angle range $\theta = [0^\circ, 120^\circ]$ and could be altered into a detector for “deep squatting” by choosing, for instance, $\theta = [0^\circ, 75^\circ]$.
- Velocity thresholds $\theta \in \mathbb{R}$ for the generic features $F_{\theta, \text{fast}}$, $F_{\theta, \text{move}}$, and $F_{\theta, \text{nmove}}$ (see Fig. 3.1 (j),(k)) tune the features’ sensitivity to the speed of motions. For small values of θ , even minute motion details will lead to a feature value of one. For larger values of θ , increasingly brisk motions are required to trigger the feature.

In general, there is no “correct” choice for a threshold. The settings depend on the specific application in mind and are left to the designer of the feature set. There will always be cases where a threshold setting will be inappropriate to characterize a certain realization of a motion. In Sect. 3.4, we will specify two manually designed feature sets that have been successfully applied in our experiments to compare the overall course of full-body motions while disregarding motion details.

To aid the designer of a feature set in choosing appropriate thresholds, it is also possible to derive threshold values from training data by supervised learning techniques. Here, one can supply a training set \mathcal{A} of “positive” motions that should yield the feature value one for most of its frames and a training set \mathcal{B} of “negative” motions that should yield the feature value zero for most of its frames. The threshold θ is then determined by a one-dimensional optimization algorithm, which iteratively maximizes the occurrences of the output one for the set \mathcal{A} while maximizing the occurrences of the output zero for the set \mathcal{B} , see [DRME06a, Dem06]. However, finding suitable positive and negative training sets turned out to be difficult in many cases since we expect most features to change their value over the course of a typical motion. For some features even, the feature value one occurs very rarely, during rather short, singular events (e. g., stretching out the hand during a punch). This necessitates very short positive training clips comprising only the relevant frames. For the optimization to make sense, the negative training clips should be chosen in such a way that the semantically “correct” threshold is barely not exceeded, which is often problematic.

As the last step of computing relational feature values, we typically apply the thresholding function H_θ to continuous values such as distances or velocities, which may be measured in different units (inches, centimeters, or arbitrary multiples thereof) depending on the specific motion capture file. To account for such differences in scale and to make motions performed

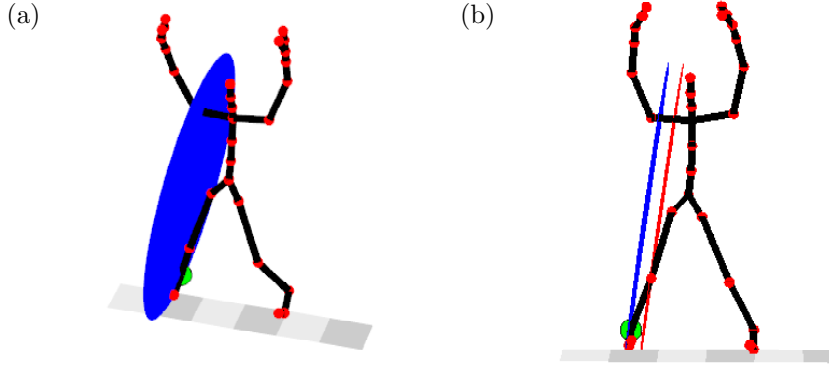


Figure 3.14. Relational feature that expresses whether the right leg is stretched sideways or not. **(a)** The feature values may randomly fluctuate if the right ankle (green dot) is located close to the decision boundary (blue disc). **(b)** Introducing a second, weaker decision boundary (position exaggerated for a better visualization) prevents the feature from fluctuations.

by different characters comparable, we express our distance and velocity thresholds θ in terms of certain constant, skeleton-defined distances, such as the length of the upper arm (humerus), which scales well with the absolute size of the skeleton. We abbreviate the resulting length unit as “hl” for “humerus length”. Similarly, we also use the relative length units “sw” and “hw”, standing for “shoulder width” and “hip width”.

3.3.3 Robust Thresholding

The simple quantization scheme using the thresholding function H_θ as described for the generic features above is prone to strong output fluctuations if the input value fluctuates slightly around the threshold. To alleviate this problem, we employ a robust quantization strategy with two thresholds: a stronger threshold, θ_1 , and a weaker threshold, θ_2 . As an example, consider the “foot sideways” detector of Fig. 3.14 (a) with the right ankle located close to the test plane, which is positioned $\theta_1 = 1.2$ hw to the right of the joint ‘rhip’. The feature will yield the output value one as soon as the ankle protrudes through the test plane.

If the actor is standing with slightly spread legs such that the right ankle is very close to the test plane, small changes of the ankle position or the hip orientation can lead to strong zero/one-fluctuations of the boolean feature value, see Fig. 3.15. By introducing the weaker threshold $\theta_2 = 1.0$ hw, which is indicated as the red plane in Fig. 3.14 and as the red line in Fig. 3.15, insignificant fluctuations can be filtered out in the following way: we only let the feature value switch over from one to zero if the distance falls below θ_2 . We refer to this strategy as *robust thresholding*, in the literature it is also known as *hysteresis thresholding* [Fau01, Chap. 4]. It turns out that this heuristic suppresses undesirable zero-one fluctuations in relational feature values very effectively, see the lower graph in Fig. 3.15.

For $\theta_1 > \theta_2$, we replace the thresholding function H_θ by the *robust thresholding operator* $H_{\theta_1, \theta_2}^{\text{robust}}$, which acts on a time-dependent sequence $x : [1 : T] \rightarrow \mathbb{R}$ as follows:

$$H_{\theta_1, \theta_2}^{\text{robust}}[x](t) := \begin{cases} 1 & \text{if } x(t) \geq \theta_1 \\ 0 & \text{if } x(t) < \theta_2 \\ H_{\theta_1, \theta_2}^{\text{robust}}[x](t-1) & \text{if } x(t) < \theta_1 \wedge x(t) \geq \theta_2 \end{cases} \quad (3.31)$$

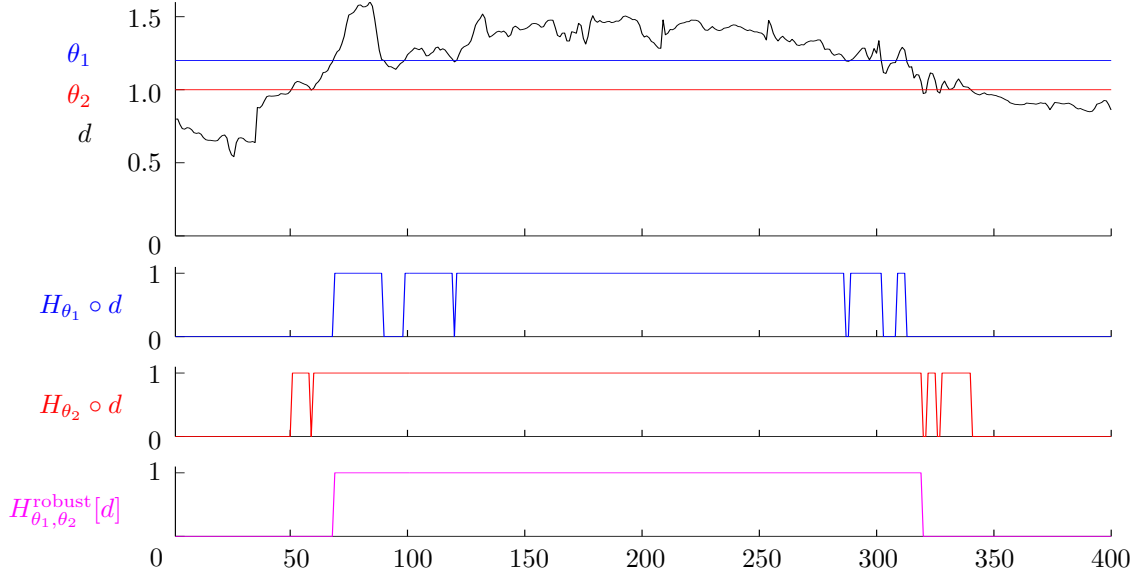


Figure 3.15. Top: Distance d of the joint ‘rankle’ to the plane that is parallel to the blue plane shown in Fig. 3.14 but passes through the joint ‘hip’, expressed in the relative length unit “hip width” (hw). The underlying motion is a Tai Chi move in which the actor is standing with slightly spread legs. The red and blue horizontal lines at $\theta_2 = 1$ hw and $\theta_1 = 1.2$ hw, respectively, indicate the two thresholds, corresponding to the red and blue planes of Fig. 3.14 (b). **Middle:** Thresholded distance signals (blue and red) using the stronger threshold, θ_1 , and the weaker threshold, θ_2 , respectively. **Bottom:** Thresholded distance signal using robust thresholding with both θ_1 and θ_2 (magenta).

where $t \in [1 : T]$ and $H_{\theta_1, \theta_2}^{\text{robust}}[x](0) := H_{\theta_1}(x(1))$. A similar robust thresholding operator can replace the characteristic function χ_θ for the case of angle ranges $\theta \subseteq [0, \pi]$.

Experiment. For a quantitative evaluation of robust thresholding, we segmented a 60-minute excerpt from the CMU mocap database [CMU03], which we refer to as $\mathcal{D}_{60}^{\text{CMU}}$. We used two versions of a combined feature function comprising the 11 relational features E_{13} to E_{23} (focusing on the lower extremities) of Tab. 3.1. Each feature exists in a nonrobust version and in a robust version, giving rise to the combined feature functions $F^{\text{standard}} : \mathcal{P} \rightarrow \{0, 1\}^{11}$ and $F^{\text{robust}} : \mathcal{P} \rightarrow \{0, 1\}^{11}$. The total length of $\mathcal{D}_{60}^{\text{CMU}}$ was 499,416 frames sampled at 120 Hz. The F^{standard} -feature sequence for $\mathcal{D}_{60}^{\text{CMU}}$ consisted of 24,078 segments with an average length of 20.7 frames ($\sigma = 131.2$ frames), whereas the F^{robust} -feature sequence consisted of 18,715 segments with an average length of 26.7 frames ($\sigma = 177.3$ frames). The maximum segment length was 6,525 frames for the standard feature function vs. 11,975 segments for the robust feature function. This shows the tendency of growing longer segments with the robust version. Fig. 3.16 gives a histogram of segment lengths for the two versions. The blue histogram corresponds to F^{standard} , while the purple/magenta overlaid histogram corresponds to F^{robust} . Up to segment length $\ell = 16$, there are more frames belonging to segments of the respective length for F^{standard} . For larger segment lengths, the magenta-colored parts of the bars indicate that F^{robust} leads to more frames belonging to segments of medium length than F^{standard} . Note that the histogram is only shown for segment lengths of up to 120 frames.

As a result, we note that the effect of robust thresholding is to lengthen the induced segments. The above considerations imply that this effect usually corresponds to unstable

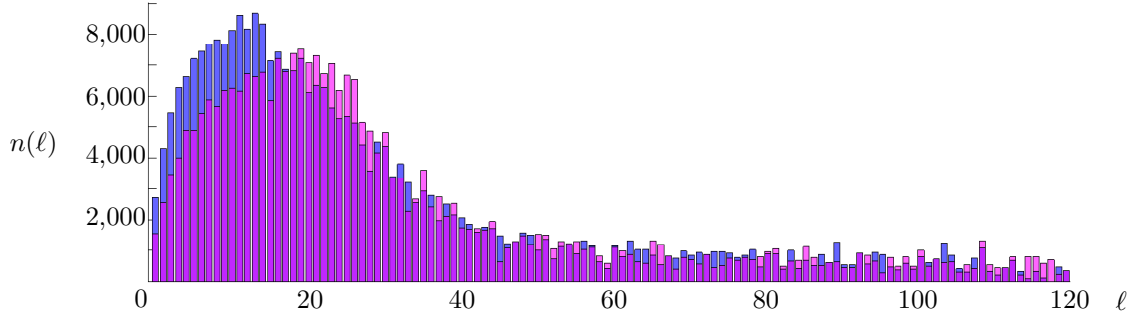


Figure 3.16. First 120 bins of histograms of segment lengths for the database $\mathcal{D}_{60}^{\text{CMU}}$ using the feature functions F^{standard} (blue histogram) and F^{robust} (magenta/purple, transparently overlaid histogram). Segment lengths ℓ run along the horizontal axis (measured in frames, 120 Hz sampling rate), while the vertical axis denotes the number of frames, $n(\ell)$, belonging to a segment of length ℓ .

segments being merged into longer, semantically meaningful segments such as the phase of “standing with spread legs” shown in Fig. 3.15.

3.3.4 Towards Automatic Feature Design

Besides learning adequate threshold values θ from training data, see Sect. 3.3.2, we also conducted experiments with automatic methods of finding sensible joint combinations for our generic relational features. In the following discussion, we exemplarily focus on the generic features of type $F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}$ and $F_{\text{nplane}}^{(j_1, j_2, j_3; j_4)}$, which depend on four joints $j_1, \dots, j_4 \in J$. Similar to the combinatorial approach of Carlsson and Sullivan [Car96, Car99, SC02] for 2D motion analysis from video data, we consider all possible test planes that can be constructed from the joints j_1, j_2, j_3 . For each of the $\binom{|J|}{3}$ 3-combinations of these joints, we test each of the $|J| - 3$ joints that have not been used in the combination against the corresponding test plane. In total, this gives us

$$(|J| - 3) \cdot \binom{|J|}{3} = 4 \cdot \binom{|J|}{4} = O(|J|^4) \quad (3.32)$$

relational features. To keep this number within reasonable bounds, we only focus on the lower extremities in our experiments, admitting the $|J| = 9$ joints ‘root’, ‘r/lhip’, ‘r/lknee’, ‘r/lankle’, and ‘r/ltoes’, which yields $4 \cdot \binom{9}{4} = 504$ different combinations. Since we want to evaluate both $F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}$ and $F_{\text{nplane}}^{(j_1, j_2, j_3; j_4)}$, the total number of features is $f := 1,008$. For a minimal representation of the entire body with $|J| = 17$, one would obtain 9,520 combinations, which turned out to be too unwieldy for the subsequent data analysis.

Two questions arise at this point. First, how does one automatically choose a meaningful threshold for each of the f features? Our approach will be to omit the thresholding step, working on raw distance signals. And second, how does one determine the quality of a feature in an automatic fashion? To find out how standard data analysis techniques perform

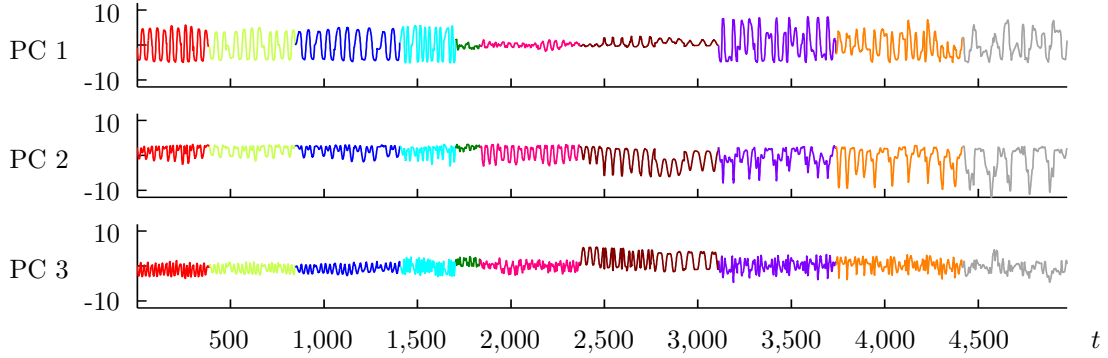


Figure 3.17. First three principal components (PC) of the training database. The different motion classes are color-coded, from left to right: walking (red), walking backwards (green), shuffling (blue), jogging (cyan), hopping on both legs (dark green), jumping jacks (pink), squats (brown), frontal kicks (purple), side kicks (orange), and cartwheels (gray).

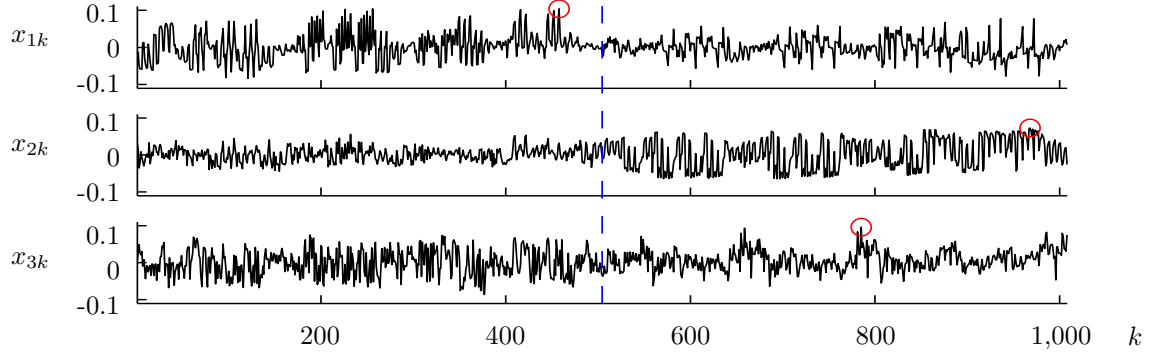


Figure 3.18. Factor loadings, x_{ik} , for the first three principal components, plotted against the feature number, k , which runs along the horizontal axis. The red circles indicate the maximum absolute factor loading for each principal component. The dashed line marks the division between features of type $F_{\text{plane}}^{(j_1, j_2, j_3; j_4)}$ (features $k = 1, \dots, 504$) and $F_{\text{nplane}}^{(j_1, j_2, j_3; j_4)}$ (features $k = 505, \dots, 1,008$).

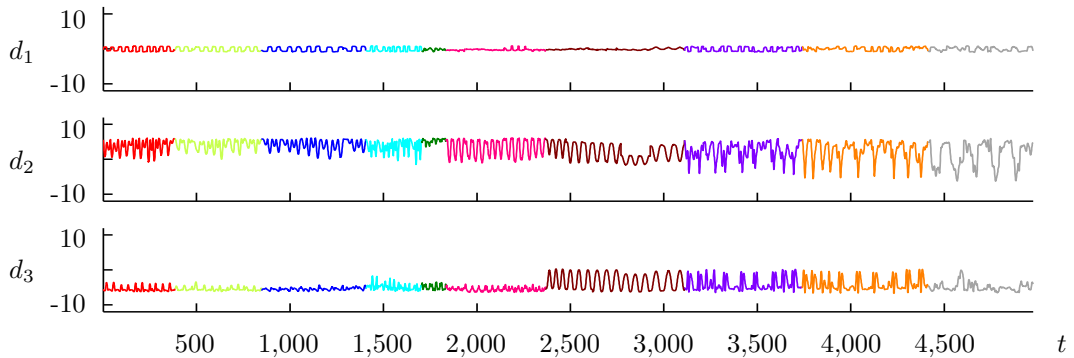


Figure 3.19. Distance signals d_i , $i = 1, 2, 3$, each corresponding to the strongest factor loading in principal component i , as marked by red circles in Fig. 3.18. Feature corresponding to d_1 : $F_{\text{plane}}^{(\text{ltoes}, \text{rhip}, \text{rtoes}; \text{lhip})}$; to d_2 : $F_{\text{nplane}}^{(\text{ltoes}, \text{rknee}, \text{rankle}; \text{lhip})}$; to d_3 : $F_{\text{nplane}}^{(\text{lhip}, \text{rknee}, \text{rankle}; \text{rhip})}$.

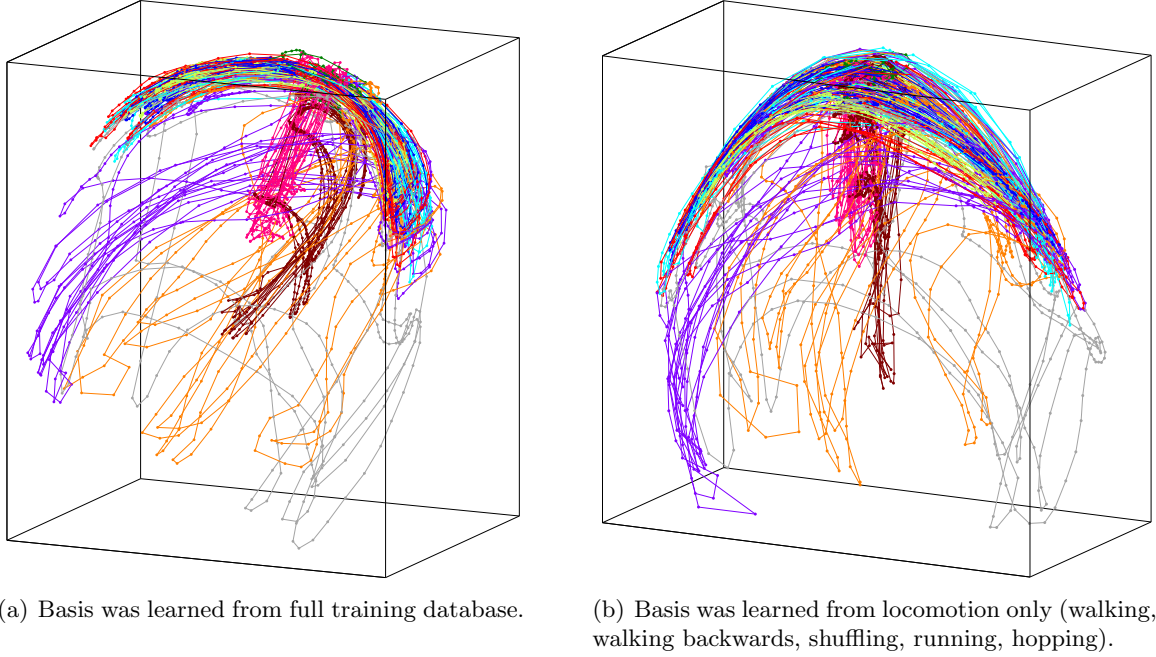


Figure 3.20. 3D plots of the first three principal components for different training databases. Despite the fact that PCA treats its input data as an unordered point set, we connected temporally adjacent data points with lines to emphasize the time-dependency of the data. As in Fig. 3.17 and Fig. 3.19, the different motion classes are color-coded as follows: walking (red), walking backwards (green), shuffling (blue), jogging (cyan), hopping on both legs (dark green), jumping jacks (pink), squats (brown), frontal kicks (purple), side kicks (orange), and cartwheels (gray).

in this setting, we decided to evaluate the features using Principal Component Analysis (PCA) [Pea01].

Given an input set of data vectors in \mathbb{R}^f , PCA finds an orthogonal basis $\mathcal{X} = \{x_1, \dots, x_f\} \subset \mathbb{R}^f$ such that the variance σ_1^2 of the data set's projection onto x_1 is maximal, the variance σ_2^2 of the projection onto x_2 is maximal within the remaining orthogonal complement, and so on. Plotting the variances σ_i^2 over i for $1 \leq i \leq f$, one typically observes an exponential drop-off in the variance [FF05]. This property can be used for dimensionality reduction by projecting the data onto the first $p < f$ basis vectors of \mathcal{X} , where the fidelity of approximation can be controlled by the choice of p . The resulting data points are p -dimensional, and each of the p dimensions is referred to as a *principal component*. In the context of statistical factor analysis [Gor83], the first p basis vectors $x_i = \sum_{k=1}^f x_{ik} e_k$, $1 \leq i \leq p$, are interpreted as uncorrelated *factors* or *latent variables* explaining the variance in the data. The vectors $e_k \in \mathbb{R}^f$ form the standard basis of \mathbb{R}^f and stand for the individual features, while the coefficients x_{ik} are referred to as *factor loadings*. Those features k for which $|x_{ik}|$ is large contribute particularly much to factor i . In that case, one also says that feature k *loads* on factor i .

For evaluation, we composed a representative training database, $D : [1 : T] \rightarrow \mathcal{P}$, which was a concatenation of 50 short motion clips comprising a total of $T = 4,975$ frames sampled at 30 Hz, or 166 seconds. The 50 motion clips were grouped into 10 motion classes, each of which was performed by 5 different actors. The motion classes were walking, walking backwards, shuffling, jogging (4 steps each), hopping on both legs (1 hop), jumping jacks,

squats (3 repetitions each), frontal kicks, side kicks (2 repetitions each), and cartwheels (1 repetition). For each pose $P = D(t)$, $t \in [1 : T]$ we then computed the distance values $d_{\text{plane}}(P^{j_1}, P^{j_2}, P^{j_3}; P^{j_4})$ for the first half of the $f = 1,008$ combinatorial features, and the distance values $d_{\text{nplane}}(P^{j_1}, P^{j_2}, P^{j_3}; P^{j_4})$ for the second half of the features (see (3.9) and (3.13), respectively.) This yields a real $f \times T$ matrix, the columns of which represent the poses of the motion and are regarded as individual data points in \mathbb{R}^f . Each row can be seen as a time-dependent distance signal, which was centered by subtracting the mean. Then, each signal was normalized by its range (the difference between the signal’s maximum and minimum), effectively equalizing the signals’ amplitudes. This was necessary since some of the signals would have larger amplitudes (and thus, higher energy—PCA would therefore assign higher factor loadings) due to “unfair” leverage effects. For example, consider the plane spanned by the fixed joints ‘lhip’, ‘root’, and ‘rhip’. Testing the joint ‘rankle’ against this plane, one obtains a distance signal of much higher amplitude than if testing the joint ‘rknee’ since the former joint is further away from the hip than the latter joint. Yet, the two distance signals express a conceptually similar feature. On the other hand, the normalization artificially amplifies low-amplitude signals that may belong to features focusing on irrelevant, small-scale motions. We had to accept this as a tradeoff, last but not least because the features of type “nplane” systematically tended to yield smaller amplitudes than those of type “plane”.

Note that we used raw distance signals as the input to the PCA algorithm, so the data differs from the actual, thresholded, relational feature values. We omitted the thresholding step because PCA does not make sense for binary data—being linear combinations of binary data, the resulting principal components would not be binary themselves. Furthermore, PCA is tailored to data with a multivariate normal distribution, which is clearly not given for binary data. Nevertheless, we also performed the above experiment with thresholded, binary data, where each distance signal was thresholded with its respective arithmetic mean. The resulting principal components were very similar as in the case of unprocessed distance data. In general, statistical data analysis for binary data is a problematic issue, to which we will return in Sect. 3.3.5.

Fig. 3.17 depicts the first $p = 3$ principal components of our data set, which together account for 66% of the variance. The quasi-periodic signals corresponding to the first four motion classes (locomotion) are in phase and therefore strongly correlated throughout the three principal components, which means that these motion classes can hardly be distinguished in the PCA-derived feature space, see Fig. 3.20 (a) for a 3D plot of the first three principal components. Here, the dense, curved, colored trajectory cluster corresponds to the four locomotion classes. The fact that walking and running are mapped to the same region in the PCA space—and would therefore be confused by a classification or retrieval algorithm based on this PCA space—is not surprising: after all, we are only focusing on the lower extremities, very similar poses occur during typical walking and running sequences, and we are disregarding the temporal aspect of the motions. This reinforces the need for additional features such as our velocity features, which succeed in separating walking from running.

On the other hand, motion classes such as jumping jacks and squats can be easily distinguished from each other since their principal components behave very differently (opposing signs, out of phase). This shows as well-separated, uniformly colored clusters in Fig. 3.20 (a). Boolean relational features corresponding to the three principal components would yield significantly different feature sequences for these motions. Again, this comes as no surprise since these motions contain specific poses that are very different from any poses occurring in other

motions. Similar findings hold for the kicking motions as well as the cartwheels. In general, the occurrence of such characteristic poses provides valuable information about a motion and often allows many candidate motion classes to be ruled out in motion classification, see also Sect. 5.2.2.

To assess the influence of the training database on the derived PCA space, we learned a second PCA basis using only the locomotion part of our database. Projecting the full database into the space spanned by the first three basis vectors of that basis, we obtained the representation shown in Fig. 3.20 (b). The dense locomotion cluster has uncoiled a bit, which reflects the way that PCA chooses the basis vectors: since most of the locomotion data is fairly similar, PCA has the liberty to focus on the small variations that distinguish different locomotion styles. For the full training database, however, too much variance introduced by other motion classes has to be captured by the first few principal components.

The principal components that we have discussed so far are linear combinations of $f = 1,008$ features, which are not efficiently manageable in practice. We therefore focus on the important features, which are those features with particularly high absolute factor loadings. The factor loadings x_{1k} , x_{2k} , and x_{3k} corresponding to the first three principal components are plotted in Fig. 3.18 for $1 \leq k \leq f$. Here, the features $k = 1, \dots, 504$ are of type “plane”, while the features $k = 505, \dots, 1,008$ are of type “nplane”. The most important features for the first three principal component were identified by PCA as

$$F_{\text{plane}}^{(\text{ltoes,rhip,rtoes;lhip})}, \quad F_{\text{nplane}}^{(\text{ltoes,rknee,rankle;lhip})}, \quad F_{\text{nplane}}^{(\text{lhip,rknee,rankle;rhip})}, \quad (3.33)$$

respectively, see the plots in Fig. 3.19. Note that up to scale, the three features are very similar to and in phase with the corresponding principal components. At first sight, these features do not have an intuitive interpretation. However, the first feature grasps the relative forward-backward movement of the legs (for instance during locomotion) by means of a plane that rotates forwards and backwards about the right hip with high amplitude. The second feature detects if the legs are spread sideways using a normal vector that contains a considerable horizontal component when the legs are moved apart from each other (‘ltoes’ to ‘rhip’). The difference between the first two features can also be seen by comparing the corresponding distance signal in the locomotion and jumping jack segments in Fig. 3.19. The third feature uses a normal vector that has a stable vertical component (‘lhip’ to ‘rknee’) and encodes the upward-downward movement of the right foot relative to the hip, which can also be seen from the third feature’s distance signal for squatting and kicking. Note that many features yield very similar factor loadings, which stems from their high correlation: for instance, replacing the joint ‘lankle’ by the joint ‘ltoes’ effectively leads to identical features.

Discussion. Automatically derived joint combinations lead to distance signals with high variance, but they are usually incapable of grasping solitary or extreme poses, which are semantically very important. Certain features that are zero for large proportions of time might be very important, for example “kick” detectors that check if a foot is raised above knee height. However, PCA would rather focus on those poses that occur most frequently in the training data, ignoring rare poses. Here, it would be necessary to apply preclustering strategies as proposed in [SKK04] to prevent frequently occurring poses from introducing a bias. Also, it has been noted in the literature that human motion, is only adequately represented with more elaborate, nonlinear models such as Local Linear Embedding (LLE), see [LE04].

Several authors such as Arikian and Forsyth [AFO03], Johnson [Joh03], or Forbes and Fiume [FF05] apply PCA for dimensionality reduction prior to comparison of motion data. Using PCA, BenAbdelkader et al. [BDC02] construct so-called Eigengait spaces for motion-based people recognition from 2D video. Our PCA-based experiment differs from these applications in that their input data is either angle-based or video-pixel-based, while we use distance values defined by our relational features. Furthermore, these applications use PCA as an online preprocessing step for their similarity measures, while our scenario takes place offline, at the feature design stage.

While providing opportunities for future research, this experiment reinforced us in having our features designed by hand: to bridge the semantic gap and provide a user with the appropriate “language” in the query formulation step of our retrieval applications, it is important to provide a rich set of intuitive, semantically interpretable features. Currently, this can only be achieved by a human expert. Nevertheless, “good”, automatically designed features might turn out to be valuable for fully automatic motion classification even if they are not intuitive. Here, quality measures based on precision/recall statistics could be helpful, see also [Dem06].

3.3.5 Statistical Feature Analysis

There are two conflicting goals in the design of a relational feature set: on the one hand, we want the resulting combined feature function to encompass as many semantically relevant aspects of a pose as possible. On the other hand, the features should exhibit as little redundancy as possible in order to keep the total number of features small and to prevent certain aspects from being overrepresented, which would introduce a bias. As an indicator of redundancy, we evaluated pairwise correlations between features. Here, the fact that our features yield binary instead of continuous data necessitates a more elaborate way of estimating correlation than provided by the common product-moment correlation. Kubinger notes in [Kub03] that the product-moment correlation tends to underestimate the true dependence between binary variables if the frequency of co-occurrence of the value one is low. As an alternative, the so-called *tetrachoric correlation* is proposed. Here, the fundamental assumption is that the binary variables are thresholded versions of some latent continuous variables, which holds for our features by construction.

The following example demonstrates the difference between product-moment and tetrachoric correlations, see also [Dem06] for computational details and further examples. We extracted the feature values for F_5 of Tab. 3.2 (rhand moving upwards) from a 39-second gymnastics sequence consisting of jumping jacks, running on the spot, squats, elbow-to-knee exercises, rotating the arms, and twisting the upper body. Furthermore, we constructed a new feature from F_5 by flipping the direction of the test plane’s normal, yielding the detector F'_5 for “rhand moving downwards”. Clearly, these two features will never assume the value one simultaneously. They are, however, closely related since the underlying continuous velocity signals are identical up to a sign. Indeed, the tetrachoric correlation between the two resulting binary features was estimated as -0.97 , while the product-moment correlation was only -0.26 . Other, less obvious correlations could also be revealed, for example a clear negative correlation between the features E_{21} (rfoot sideways) and E_{23} (feet crossed over) of Tab.3.1.

Pairwise tetrachoric correlations can point at dependencies between relational features, but they do not provide alternatives in a constructive fashion. However, simply removing one of the dependent features can often be feasible without weakening the semantic expressiveness of the feature set: for example, we decided not to include the above mentioned feature F'_5

(rhand moving downwards) in order to keep our feature set F of Tab. 3.2 small. At first sight, F could then be considered as unable to distinguish a resting hand from a hand moving downwards, since the feature F_5 (rhand moving upwards) would yield the feature value zero for both cases. But we still have the feature F_{13} (rhand fast), which will distinguish the two cases: a hand that is not moving upwards but is fast must be moving either downwards or sideways. In general, such combinations of features can grasp additional aspects of a motion by exploiting logical implications between elementary features. Adding additional features that specialize on these aspects is not necessary.

3.4 Experimental Feature Sets

The design of relational motion features expressing intuitive, semantic qualities currently requires the expertise of a human. This is the main conclusion from our experiments on PCA-based methods for feature design and tetrachoric correlation for feature analysis as well as extensive retrieval experiments with different feature sets. In designing our features by hand, we incorporated most parts of the body, in particular the end effectors, so as to create a well-balanced feature set. One guiding principle was to cover the space of possible end effector locations by means of a small set of pose-dependent space “octants” defined by three intersecting planes each (above/below, left of/right of, in front of/behind the body). This subdivision is supported by our findings from PCA-based feature design, see Sect. 3.3.4. Obviously, such a subdivision is only suitable to capture the rough course of a motion, since the feature function would often yield a constant output value for small-scaled motions. Here, the features of type $F_{\text{move}}^{(j_1, j_2, j_3; j_4)}$ and $F_{\text{nmove}}^{(j_1, j_2, j_3; j_4)}$ provide a finer view on motions by additionally considering directional information.

We used common sense and human knowledge about the typical range of motions in combination with the supervised learning technique introduced in Sect. 3.3.2 to determine sensible threshold values θ . The resulting features sufficed to prove the applicability of our concepts in the retrieval and classification scenarios. For the future, it would be desirable to back up the design process by expert knowledge from the sports sciences or cognitive psychology. Currently, however, these areas of science seem to have no theory about the role of certain body-defined planes or other relational features for human motion perception and understanding. The fundamental question of whether it is desirable to mimic human perception for fully automatic motion analysis goes beyond the scope of this thesis.

Our initial retrieval experiments (Chap. 4) were performed with the feature set E consisting of $e = 31$ relational features, see Tab. 3.1. Encoding relations within the upper part of the body, relations within the lower part of the body, and interactions between the two parts, the feature set is divided into the subsets “upper”, “lower”, and “mix”, which are abbreviated as u , ℓ and m , respectively. Features with two entries in the ID column exist in two versions pertaining to the right/left half of the body but are only described for the right half—the features for the left half can be easily derived by symmetry. For the feature set E , we did not yet use robust thresholding, thus necessitating only one threshold parameter, θ .

Later, we switched to the modified feature set F consisting of $f = 39$ relational features, see Tab. 3.1. We removed the features of type $F_{\theta, \text{touch}}^{(j_1, j_2; j_3)}$, since they were too specific and their semantics were often not clear: for example, the features E_{26}/E_{27} (hands touching head) would often yield the value one coincidentally, and increasing the threshold would

ID	set	type	j_1	j_2	j_3	j_4	θ	description	
E_1/E_2	u	F_{nplane}	root	lshoulder	rshoulder	rwrist	1.0 hl	rhand in front	
E_3/E_4	u	F_{nplane}	chest	neck	neck	rwrist	0.0 hl	rhand above neck	
E_5/E_6	u	F_{nplane}	lshoulder	rshoulder	rshoulder	rwrist	1.0 sw	rhand reaching sideways	
E_7/E_8	u	F_{angle}	relbow	rshoulder	relbow	rwrist	$[0^\circ, 120^\circ]$	relbow bent	
E_9/E_{10}	u	F_{fast}	rwrist				2.5 hl/s	rhand fast	
E_{11}	u	F_{nplane}	Plane Π fixed at lshoulder, normal rshoulder \rightarrow lshoulder. Test: rwrist closer to Π than lwrist?						hands crossed over
E_{12}	u	F_{touch}	$F_{\theta, \text{touch}}^{\text{rfingers, rwrist; lfingers} \vee F_{\theta, \text{touch}}^{\text{rfingers, rwrist; lwrist}}}$ $F_{\theta, \text{touch}}^{\text{lfingers, lwrist; rfingers} \vee F_{\theta, \text{touch}}^{\text{lfingers, lwrist; rwrist}}}$					0.4 hl	hands touching
E_{13}/E_{14}	ℓ	F_{plane}	root	lhip	ltoes	rankle	0 hl	rfoot behind lleg	
E_{15}/E_{16}	ℓ	F_{nplane}	$(0, 0, 0)^\top$	$(0, 1, 0)^\top$	root	rankle	-1.8 hl	rfoot raised	
E_{17}/E_{18}	ℓ	F_{fast}	$F_{\theta, \text{fast}}^{\text{rankle}} \wedge F_{\theta, \text{fast}}^{\text{rtoes}}$					2.5 hl/s	rfoot fast
E_{19}/E_{20}	ℓ	F_{angle}	rknee	rhip	rknee	rankle	$[0^\circ, 120^\circ]$	rknee bent	
E_{21}/E_{22}	ℓ	F_{nplane}	lhip	rhip	rhip	rankle	1.0 hw	rfoot sideways	
E_{23}	ℓ	F_{nplane}	Plane Π fixed at lhip, normal rhip \rightarrow lhip. Test: rankle closer to Π than lankle?						feet crossed over
E_{24}/E_{25}	m	F_{touch}	$F_{\theta, \text{touch}}^{\text{rknee, rankle; rfingers} \vee F_{\theta, \text{touch}}^{\text{rknee, rankle; rwrist} \vee F_{\theta, \text{touch}}^{\text{rankle, rtoes; rfingers}}}$ $F_{\theta, \text{touch}}^{\text{rankle, rtoes; rwrist} \vee F_{\theta, \text{touch}}^{\text{lknee, lankle; rfingers} \vee F_{\theta, \text{touch}}^{\text{lknee, lankle; rwrist}}}$ $F_{\theta, \text{touch}}^{\text{rankle, rtoes; rfingers} \vee F_{\theta, \text{touch}}^{\text{rankle, rtoes; rwrist}}}$					1 hl	rhand touching either leg
E_{26}/E_{27}	m	F_{touch}	neck	headtop	rfingers		1 hl	rhand touching head	
E_{28}/E_{29}	m	F_{touch}	rhip	lhip	rfingers		1 hw	rhand touching hips	
E_{30}	m	F_{angle}	$F_{\theta, \text{angle}}^{\text{root, neck; rhip, rknee}} \wedge F_{\theta, \text{angle}}^{\text{root, neck; lhip, lknee}}$					$[0^\circ, 120^\circ]$	torso bent
E_{31}	m	F_{fast}	root				1.0 hl/s	root fast	

Table 3.1. The initial feature set, E , consisting of $e = 31$ relational features. The abbreviations “hl”, “sw”, and “hw” denote the relative length units “humerus length”, “shoulder width”, and “hip width”, respectively.

ID	set	type	j_1	j_2	j_3	j_4	θ_1	θ_2	description	
F_1/F_2	u	F_{move}	neck	rhip	lhip	rwrist	1.8 hl/s	1.3 hl/s	rhand moving forwards	
F_3/F_4	u	F_{nplane}	chest	neck	neck	rwrist	0.2 hl	0 hl	rhand above neck	
F_5/F_6	u	F_{move}	belly	chest	chest	rwrist	1.8 hl/s	1.3 hl/s	rhand moving upwards	
F_7/F_8	u	F_{angle}	relbow	rshoulder	relbow	rwrist	$[0^\circ, 110^\circ]$	$[0^\circ, 120^\circ]$	relbow bent	
F_9	u	F_{nplane}	lshoulder	rshoulder	lwrist	rwrist	2.5 sw	2 sw	hands far apart, sideways	
F_{10}	u	F_{move}	lwrist	rwrist	rwrist	lwrist	1.4 hl/s	1.2 hl/s	hands approaching each other	
F_{11}/F_{12}	u	F_{move}	rwrist	root	lwrist	root	1.4 hl/s	1.2 hl/s	rhand moving away from root	
F_{13}/F_{14}	u	F_{fast}	rwrist				2.5 hl/s	2 hl/s	rhand fast	
F_{15}/F_{16}	ℓ	F_{plane}	root	lhip	ltoes	rankle	0.38 hl	0 hl	rfoot behind lleg	
F_{17}/F_{18}	ℓ	F_{nplane}	$(0, 0, 0)^\top$	$(0, 1, 0)^\top$	$(0, y_{\min}, 0)^\top$	rankle	1.2 hl	1 hl	rfoot raised	
F_{19}	ℓ	F_{nplane}	lhip	rhip	lankle	rankle	2.1 hw	1.8 hw	feet far apart, sideways	
F_{20}/F_{21}	ℓ	F_{angle}	rknee	rhip	rknee	rankle	$[0^\circ, 110^\circ]$	$[0^\circ, 120^\circ]$	rknee bent	
F_{22}	ℓ	F_{nplane}	Plane Π fixed at lhip, normal rhip \rightarrow lhip. Test: rankle closer to Π than lankle?						feet crossed over	
F_{23}	ℓ	F_{move}	Consider velocity v of rankle relative to lankle in rankle \rightarrow lankle direction. Test: projection of v onto rhip \rightarrow lhip line large?						feet moving towards each other, sideways	
F_{24}	ℓ	F_{move}	Same as above, but use lankle \rightarrow rankle instead of rankle \rightarrow lankle direction.						feet moving apart, sideways	
F_{25}/F_{26}	ℓ	F_{fast}	$F_{\theta, \text{fast}}^{\text{rankle}} \wedge F_{\theta, \text{fast}}^{\text{rtoes}}$					2.5 hl/s	2 hl/s	rfoot fast
F_{27}/F_{28}	m	F_{angle}	neck	root	rshoulder	relbow	$[25^\circ, 180^\circ]$	$[20^\circ, 180^\circ]$	rhumus abducted	
F_{29}/F_{30}	m	F_{angle}	neck	root	rhip	rknee	$[50^\circ, 180^\circ]$	$[45^\circ, 180^\circ]$	rfemur abducted	
F_{31}	m	F_{plane}	rankle	neck	lankle	root	0.5 hl	0.35 hl	root behind frontal plane	
F_{32}	m	F_{angle}	neck	root	$(0, 0, 0)^\top$	$(0, 1, 0)^\top$	$[70^\circ, 110^\circ]$	$[60^\circ, 120^\circ]$	spine horizontal	
F_{33}/F_{34}	m	F_{nplane}	$(0, 0, 0)^\top$	$(0, -1, 0)^\top$	$(0, y_{\min}, 0)^\top$	rwrist	-1.2 hl	-1.4 hl	rhand lowered	
F_{35}/F_{36}	m	F_{plane}	Plane Π through rhip, lhip, neck. Test: rshoulder closer to Π than lshoulder?						shoulders rotated right	
F_{37}	m		Test: y_{\min} and y_{\max} close together?						y -extents of body small	
F_{38}	m		Project all joints onto xz -plane. Test: diameter of projected point set large?						xz -extents of body large	
F_{39}	m	F_{fast}	root				2.3 hl/s	2 hl/s	root fast	

Table 3.2. The current feature set, F , consisting of $f = 39$ relational features. The abbreviations “hl”, “sw”, and “hw” denote the relative length units “humerus length”, “shoulder width”, and “hip width”, respectively.

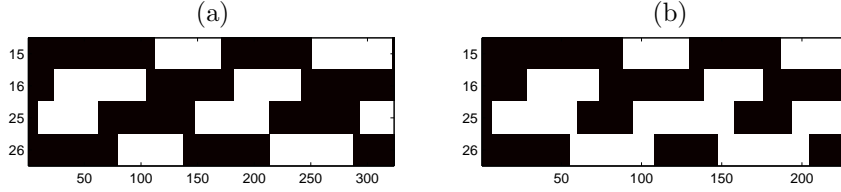


Figure 3.21. Relational feature matrices for (a) “walking” and (b) “jogging” using $f = 4$ features. The matrices are color coded as black (0) and white (1), and the numbers in front of the rows refer to Tab. 3.2. Time (measured in frames) runs along the horizontal axis.

prevent many instances of actually touching one’s head from being recognized. In part, this was due to problems in the mocap data, where faulty skeleton fitting would lead to poses where the hands are supposed to touch the head but end up being separated from the head by more than two head’s diameters. Instead, we added improved detectors for the global pose to the new feature set: F_{31} (root behind frontal plane), F_{32} (spine horizontal), F_{37} (y -extents of body small), and F_{38} (xz -extents of body large). We also incorporated some features of type $F_{\text{move}}^{(j_1, j_2, j_3; j_4)}$ and $F_{\text{nmove}}^{(j_1, j_2, j_3; j_4)}$. Absolute coordinates, as used in the definition of features such as F_{17} , F_{32} , or F_{33} , stand for virtual joints at constant 3D positions w. r. t. an $(x, y, z)^\top$ world system in which the y axis points upwards. The symbols y_{\min}/y_{\max} denote the minimum/maximum y coordinates assumed by the joints of a pose that are not tested. Features such as F_{22} do not follow the same derivation scheme as the other features and are therefore described in words.

In general, all of our relational features are defined in such a way that the feature value zero corresponds to a neutral, standing pose. If a feature assumes the value one, something “extraordinary” that is related to the intended semantics of the feature is happening. In order to match this convention, the features E_{13}/E_{14} and F_{15}/F_{16} (“r/lfoot *behind* l/rleg”) have been given the opposite semantics of Fig. 3.1 (a) and the previous explanations, where the features yielded the value one for the states “r/lfoot *in front of* l/rleg”.

3.5 Measuring Motion Similarity with Relational Features

3.5.1 A Simple Example: Walking vs. Jogging

To compare the walking and the running motion of Fig. 2.1, we consider the feature function F^{legs} consisting of the $f = 4$ features F_{15}/F_{16} (“r/lfoot *behind* l/rleg”) and F_{25}/F_{26} (“r/lfoot *fast*”). Denoting two mocap data streams by D and D' (lengths: N and M frames, respectively), we compute the feature sequences $X := F^{\text{legs}} \circ D$ and $X' := F^{\text{legs}} \circ D'$, which we think of as *feature matrices* $X \in \{0, 1\}^{f \times N}$ and $X' \in \{0, 1\}^{f \times M}$, see Fig. 3.21. We use the functional notation $X(n)$ to refer to the n^{th} column of X for $n \in [1 : N]$. A comparison of Fig. 3.21 (a) and (b) reveals that the first two rows of the feature matrices are identical in structure for the two motions. This corresponds to the fact that both walking and jogging motions are dominated by an alternation of the condition “left/right foot in front”. The last two rows differ by the occurrence of phases where both feet are simultaneously fast: these are exactly the air phases of the jogging motion, see frames 50, 100, 150, and 200 of Fig. 3.21 (b).

Obviously, it is possible to determine the notion of similarity by selecting appropriate features from the global feature set: based on their feature matrices, walking and jogging

would be considered as similar if only the first two features were used, and as dissimilar if the full feature function F^{legs} was used.

3.5.2 Cost Matrices Based on Relational Features

Next, we construct a local distance measure based on relational features and compare the resulting cost matrices to the quaternion distance and the 3D point cloud distance as introduced in Sect. 2.5. As usual, we assume a fixed feature function $F : \mathcal{P} \rightarrow \{0, 1\}^f$. Given two mocap data streams D and D' of lengths N and M , respectively, we compute their feature matrices $X := F \circ D$ and $X' := F \circ D'$. Individual boolean feature vectors are then compared using the Hamming distance (counting the number of disagreements) as the local distance measure. For $n \in [1 : N]$ and $m \in [1 : M]$, we define

$$c^{\text{rel}}(n, m) := \frac{1}{f} \sum_{i=1}^f |X(n)_i - X'(m)_i|, \quad (3.34)$$

where $X(n)_i \in \{0, 1\}$ denotes the i^{th} entry of the feature vector $X(n)$, and $X'(m)_i$ is defined similarly. Note that the Hamming distance coincides with the Manhattan (ℓ^1) distance in the case of boolean vectors.

Using the feature set F of Tab. 3.2 comprising $f = 39$ relational features, we computed the cost matrices C^{rel} for the example motions of Sect. 2.8. The results are shown in Fig. 3.22–3.23. Comparing the two walking motions of Fig. 2.7 (a) using c^{rel} yields the cost matrix of Fig. 3.22 (a), which exhibits more or less the same structure as the cost matrices of Fig. 2.12, while the regions of high dissimilarity are not as dominant. A major difference is the apparent block structure of C^{rel} , which corresponds to the constant runs of feature vectors forming the F -segmentation of the mocap data streams.

The cost matrix of Fig. 3.22 (b) for “rotating both arms forwards” is shown along with an optimal warping path. The repetitive structure of the motions is as clear as in the cost matrix $C^{3\text{D}}$ of Fig. 2.13 (b). Also, the warping path is very similar to the warping path of Fig. 2.13 (b) (up to the jagged look that is due to the discrete nature of relational features) and reflects the true underlying time warp. This is in contrast to the quaternion-based warping path, which is a rather poor approximation of the true time warp. Similar findings hold for the cost matrices of Fig. 3.23. The ballet example of Fig. 3.23 (c) shows the high degree of noise tolerance of c^{rel} compared to the quaternion distance.

As a final example, we consider the cost matrices of Fig. 3.24, which correspond to the comparison of “climbing stairs” and “walking”. The cost matrix of Fig. 3.24 (a) does not exhibit a clear alignment path near the diagonal. This is due to significant differences in certain features such as “knees bent” or “feet raised”, as well as different arm movements. Restricting the feature function to F^{legs} leads to the cost matrix of Fig. 3.24 (b), which shows a very clear diagonal structure providing an alignment of the two motions.

In summary, relational features provide a good basis for a global similarity measure based on DTW that can be computed much more efficiently than, for example, the 3D point cloud distance. Also, the distance function c^{rel} is very robust to noise due to the strong quantization of relational features. As a further benefit, it is possible to focus on certain aspects of the motions by suitably choosing the underlying relational feature function, F . Yet, even without a restriction of the feature function, c^{rel} works well for a large range of motions. In Sect. 4.3.1

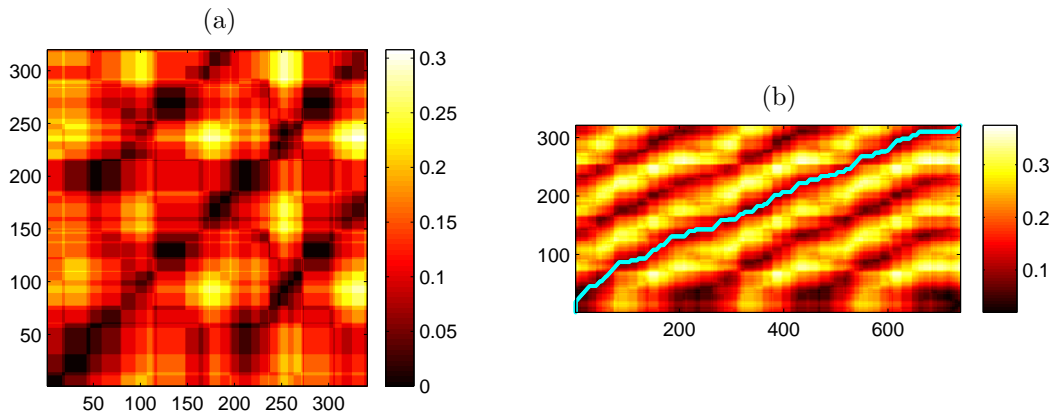


Figure 3.22. Cost matrices for (a) “walking” and (b) “rotating both arms forwards while walking” based on relational features. Compare these figures to Fig. 2.12 and Fig. 2.13, respectively.

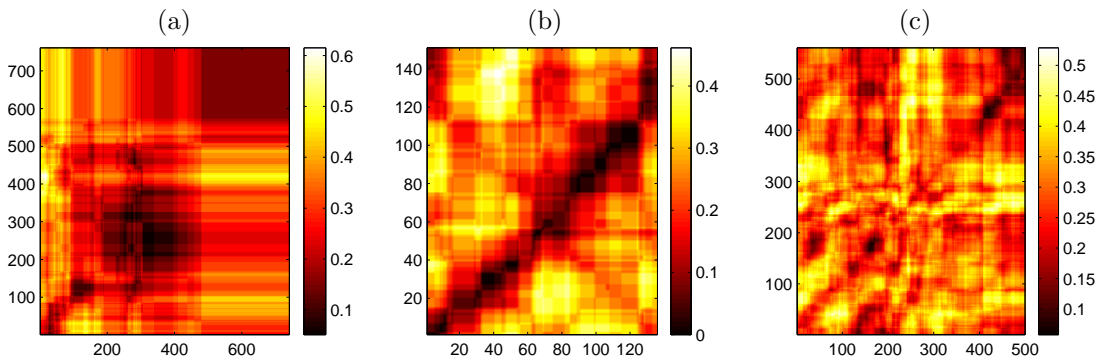


Figure 3.23. Cost matrices for (a) “lying down”, (b) “jumping jack”, and (c) “ballet” based on relational features. Compare these figures to Fig. 2.15, Fig. 2.16, and Fig. 2.17, respectively.

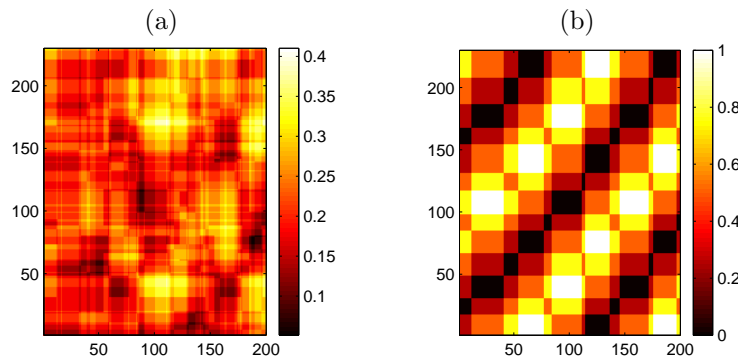


Figure 3.24. Cost matrices for “climbing stairs” based on relational features using (a) the full feature set F and (b) the restricted feature function F^{legs} . Compare these figures to Fig. 2.14.

and Sect. 5.2, we will see how c^{rel} and variants can be used as a ranking function to postprocess hits for motion queries.

Chapter 4

Efficient Index-Based Motion Retrieval

4.1 Query and Hit Concepts

Given a relational feature function $F : \mathcal{P} \rightarrow \{0, 1\}^f$, we reduce the problem of motion retrieval to a string matching problem over the alphabet $\{0, 1\}^f$. To this end, we identify a motion capture data stream with its F -feature sequence while storing the time duration in frames for each of the constituting segments. We can then reconstruct the frame range corresponding to any segment range within the F -feature sequence. We will now systematically introduce several concepts of motion *queries* and *hits* for such queries, where each successive concept adds further degrees of fault tolerance over the preceding concept. For further material related to the contents of this chapter, we refer to [MRC05b] and to our web sites [MRC05a, DRME06b], which also host the video that can be found on the accompanying CD-ROM.

The following notation is used throughout the subsequent sections. We are given a database consisting of a collection $\mathcal{D} = (D_1, D_2, \dots, D_I)$ of motion capture data streams D_i , $i \in [1 : I]$. To simplify things, we may assume that \mathcal{D} consists of one large document D by concatenating the documents D_1, \dots, D_I while keeping track of document boundaries in a supplemental data structure. Fixing a feature function $F : \mathcal{P} \rightarrow \{0, 1\}^f$, we use the notation $F[D] = \vec{w} = (w_0, w_1, \dots, w_M)$ to denote the resulting F -feature sequence of D .

4.1.1 Exact Queries and Exact Hits

One possible way of formulating a motion query is to simply specify a feature sequence $\vec{v} = (v_0, v_1, \dots, v_N)$, which we will refer to as an *exact query*. Then, an *exact hit* for \vec{v} in the database feature sequence $\vec{w} = (w_0, w_1, \dots, w_M)$ is an element $k \in [0 : M]$ such that \vec{v} is a subsequence of consecutive feature vectors in \vec{w} starting from index k . In symbols, we write $\vec{v} \sqsubset_k \vec{w}$, where

$$\vec{v} \sqsubset_k \vec{w} \quad :\Leftrightarrow \quad \forall i \in [0 : N] : v_i = w_{k+i}. \quad (4.1)$$

The set of all exact hits for \vec{v} in the database \mathcal{D} is defined as

$$H_{\mathcal{D}}(\vec{v}) := \{k \in [0 : M] \mid \vec{v} \sqsubset_k \vec{w}\}. \quad (4.2)$$

As an illustration, we search our example document D_{walk} for the exact query $\vec{v}_{\text{walk},1} = ((\binom{1}{0}), (\binom{1}{1}), (\binom{0}{1}))$ using the feature function F^{walk} . Fig. 4.1 shows the two resulting exact hits, which start with the third and seventh element of $F^{\text{walk}}[D_{\text{walk}}]$, respectively, hence $H_{\mathcal{D}_{\text{walk}}}(\vec{v}_{\text{walk},1}) = \{3, 7\}$. Feature sequences are color-coded, and hits are visualized as copies

of the query that are horizontally aligned with the database feature sequence at the respective hit position, k . Here, each hit corresponds to a right/left step sequence, cf. Fig. 3.7.

The disadvantage of the exact hit concept is that a single mismatching segment can rule out a hit even if all the other segments match, leading to a false dismissal. This can be seen in Fig. 4.2, where we search for the query $\vec{v}_{\text{jump},1} = ((\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}), (\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}), (\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}), (\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}), (\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}))$ in the document D_{jump} as well as in a variation D'_{jump} using the feature function F^{jump} . The document D'_{jump} has been obtained from D_{jump} by slightly modifying the motion in such a way that not the left leg is bent first during the time period corresponding to segment number one, but the right leg. Note that this modification does not change the overall semantics of the motion, but it changes the feature vector of segment number one from $(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix})$ to $(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix})$. Consequently, we obtain only one hit for the exact query $\vec{v}_{\text{jump},1}$ in D'_{jump} , while we obtain two hits in the original sequence D_{jump} . The first hit corresponds to the sequence “standing with parallel legs—building up momentum by bending knees—pushing off by stretching knees”. The second hit corresponds to the sequence “stretched knees during flight phase—bending knees during landing—standing with parallel legs”. Recall from Fig. 3.10 that segments number 1, 3, 5, and 7 are *transitory segments* of very short duration. They assume one of the feature values $(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix})$ or $(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix})$ more or less randomly, depending on which of the knees is bent/stretched first during the push off and landing phases. In our example, the feature vectors $(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix})$ arise because the actor has a tendency to keep the right leg bent a bit longer than the left leg. There are many possible transitions from, e.g., $(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix})$ (legs bent) to $(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix})$ (legs stretched), such as $(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 0 \\ 0 \end{smallmatrix})$, $(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 0 \\ 0 \end{smallmatrix})$, or $(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}) \rightarrow (\begin{smallmatrix} 0 \\ 0 \end{smallmatrix})$. Therefore, it is likely that the feature sequences corresponding to other instances of jumping motions differ from \vec{w} at segments number 1, 3, 5, and 7, even if the two motions are very similar.

This example demonstrates a situation where our feature concept systematically fails to provide the desired invariance under spatio-temporal deformations. If a motion inherently contains aspects that change nearly simultaneously, such as bent/stretched knees during a parallel-leg jump, one typically observes unstable, transitory segments of short duration because it is very improbable that the aspects change during the exact same frame if we assume typical sampling rates of up to 120 Hz. To address this problem and to enhance the overall robustness of the retrieval, we will now extend the matching strategy by additional fault tolerance mechanisms.

4.1.2 Fuzzy Queries and Fuzzy Hits

The concept of fuzzy queries allows the user to express uncertainty about parts of the query by admitting a whole set of possible, alternative feature vectors at each position in the query sequence. This is modeled as follows. A *fuzzy set* is a subset $V \subset \{0, 1\}^f$. Then a *fuzzy query* is defined to be a sequence $\vec{V} = (V_0, V_1, \dots, V_N)$ of fuzzy sets. Extending the definition in (4.1), a *fuzzy hit* is an element $k \in [0 : M]$ such that $\vec{V} \sqsubset_k \vec{w}$, where

$$\vec{V} \sqsubset_k \vec{w} \quad :\Leftrightarrow \quad \forall i \in [0 : N] : V_i \ni w_{k+i}. \quad (4.3)$$

Obviously, the case $V_i = \{v_i\}$ for $0 \leq i \leq N$ reduces to the case of an exact hit. Similar to (4.2), the set of all fuzzy hits for \vec{V} in \mathcal{D} is defined as

$$H_{\mathcal{D}}(\vec{V}) := \{k \in [0 : M] \mid \vec{V} \sqsubset_k \vec{w}\}. \quad (4.4)$$

Consider Fig. 4.3 for an illustration of the fuzzy concept. Here, we search the document D_{walk} for the fuzzy query $\vec{V}_{\text{walk},2} = (V_0, V_1, V_2)$ with $V_0 = V_2 = \{(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}), (\begin{smallmatrix} 1 \\ 0 \end{smallmatrix})\}$ and $V_1 = \{(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix})\}$ using

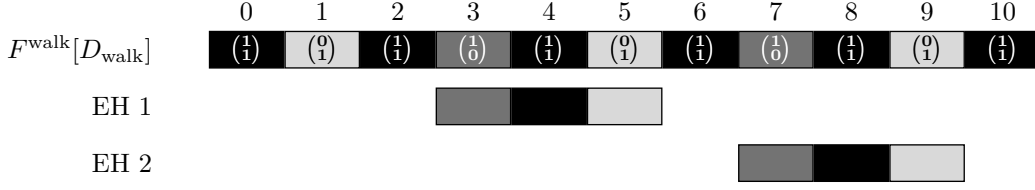


Figure 4.1. Upper row: feature sequence $F^{\text{walk}}[D_{\text{walk}}]$. **Below:** two exact hits (EH) for $\vec{v}_{\text{walk},1}$ in $F^{\text{walk}}[D_{\text{walk}}]$, indicated by copies of $\vec{v}_{\text{walk},1}$ that are horizontally aligned with $F^{\text{walk}}[D_{\text{walk}}]$ at the matching positions. In our notation, the hits read as $\vec{v}_{\text{walk},1} \sqsubset_3 F^{\text{walk}}[D_{\text{walk}}]$ and $\vec{v}_{\text{walk},1} \sqsubset_7 F^{\text{walk}}[D_{\text{walk}}]$.

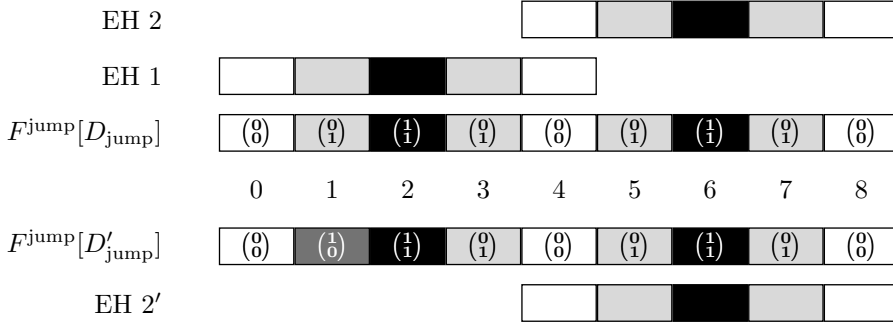


Figure 4.2. Middle rows, with feature vectors: feature sequences $F^{\text{jump}}[D_{\text{jump}}]$ and $F^{\text{jump}}[D'_{\text{jump}}]$. **Above:** two exact hits (EH) for $\vec{v}_{\text{jump},1}$ in $F^{\text{jump}}[D_{\text{jump}}]$. **Below:** the only exact hit in $F^{\text{jump}}[D'_{\text{jump}}]$.

the feature function F^{walk} . There are four fuzzy hits starting at positions 1, 3, 5, and 7, respectively. Due to the admission of the fuzzy sets $\left\{\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right\}$ at the first and third position of the fuzzy query, both left/right and right/left step sequences can be found. Compare this to the exact query in Fig. 4.1, where only right/left step sequences were found. In this example, the fuzzy concept was employed to deliberately leave certain aspects of the query unspecified, namely which foot should be in the front at the boundaries of the desired hits.

In Fig. 4.4, we search the document D'_{jump} for the fuzzy query $\vec{V}_{\text{jump},2} = (V_0, V_1, V_2, V_3, V_4)$ with $V_0 = V_4 = \left\{\begin{pmatrix} 0 \\ 0 \end{pmatrix}\right\}$, $V_1 = V_3 = \left\{\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right\}$, and $V_2 = \left\{\begin{pmatrix} 1 \\ 1 \end{pmatrix}\right\}$ using the feature function F^{jump} . Comparing the resulting fuzzy hits to the exact hits in Fig. 4.2, it turns out that both intended hits are now found. This shows that a further possible application of the fuzzy concept is to mask out transitory segments by admitting all feature vectors that may occur at the respective positions in the query.

4.1.3 Adaptive Fuzzy Hits

The concept of fuzzy hits as introduced above still lacks an important degree of flexibility: so far, the fuzziness only refers to the spatial domain (admitting alternative choices for the pose-based features) but does not take the temporal domain into account. More precisely, the segmentation of the document D is only determined by the feature function F , disregarding the fuzziness of the query \vec{V} . Here, it would be desirable if the matching strategy allowed for a fuzzy set to match with multiple successive elements of \vec{w} . For example, considering the fuzzy query $\vec{V}_{\text{walk},3} = (V_0, V_1, V_2)$ with $V_0 = V_2 = \left\{\begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$ and $V_1 = \left\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right\}$, one easily checks in Fig. 4.3 that the set of fuzzy hits, $H_{\mathcal{D}_{\text{walk}}}(\vec{V})$, is empty. By contrast, one obtains two hits for \vec{V} using the concept of *adaptive fuzzy search*, see Fig. 4.5. Note that the fuzzy

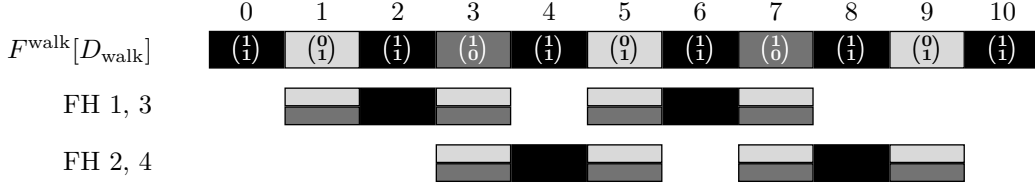


Figure 4.3. Upper row: feature sequence $F^{\text{walk}}[D_{\text{walk}}]$. **Below:** four fuzzy hits (FH) for $\vec{V}_{\text{walk},2}$ in $F^{\text{walk}}[D_{\text{walk}}]$. Fuzzy sets are represented by vertically stacked boxes of the respective color.

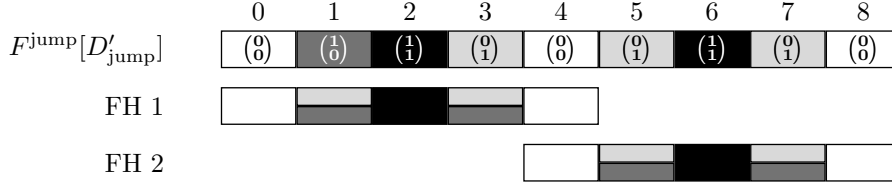


Figure 4.4. Upper row: feature sequence $F^{\text{jump}}[D'_{\text{jump}}]$. **Below:** two fuzzy hits (FH) for $\vec{V}_{\text{jump},2}$ in $F^{\text{jump}}[D'_{\text{jump}}]$.

set V_1 is matched with the contiguous segment ranges 2–4 and 6–8, respectively. The two hits correspond to left/right/left step sequences.

The general strategy is to adjust the temporal segmentation of D to the fuzziness of the query *during the matching* as follows: supposing $w_k \in V_0$ with $w_{k-1} \notin V_0$ for some index $k_0 := k \in [0 : M]$, we determine the maximal index $k_1 \geq k_0$ with $w_m \in V_0$ for all $m = k_0, k_0 + 1, \dots, k_1 - 1$ and concatenate all segments corresponding to these w_m into one large segment. By construction, $w_{k_1} \notin V_0$. Only if $w_{k_1} \in V_1$, we proceed in the same way, determining some maximal index $k_2 > k_1$ with $w_m \in V_1$ for all $m = k_1, k_1 + 1, \dots, k_2 - 1$, and so on. In case we find a sequence of indices $k_0 < k_1 < \dots < k_N$ constructed iteratively in this fashion we say that $k \in [0 : M]$ is an *adaptive fuzzy hit* and write $\vec{V} \sqsubset_k^{\text{ad}} \vec{w}$. The set of all adaptive fuzzy hits for \vec{V} in \mathcal{D} is given by

$$H_{\mathcal{D}}^{\text{ad}}(\vec{V}) := \{k \in [0 : M] \mid \vec{V} \sqsubset_k^{\text{ad}} \vec{w}\}. \quad (4.5)$$

In view of this matching technique, we return to our example $D = D_{\text{walk}}$ and $F = F^{\text{walk}}$ with the fuzzy query $\vec{V}_{\text{walk},3}$, see Fig. 4.5. We obtain $k_0 = 1, k_1 = 2, k_3 = 5$ for the first hit and $k_0 = 5, k_1 = 6, k_2 = 9$ for the second hit. In the case of the first hit, for example, this means that V_0 corresponds to segment 1 of \vec{w} , V_1 to segments 2–4, and V_2 to segment 5, amounting to a coarsened segmentation of D . In this particular example, adaptive fuzzy search can also be understood as simulating exact search on D with respect to a restricted version of the feature function F^{walk} : only the feature F^r is relevant to the segmentation. In general, adaptive fuzzy search enables a user to mask out some of the f components of the feature function F to obtain a less restrictive search leading to more hits, see Sect. 4.3.

In Fig. 4.6, we search the document D'_{jump} for the fuzzy query $\vec{V}_{\text{jump},3} = (V_0, V_1, V_2)$ with $V_0 = V_2 = \{\begin{pmatrix} 0 \\ 0 \end{pmatrix}\}$ and $V_1 = \{\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\}$ using the feature function F^{jump} . The adaptive fuzzy hits are the same as the fuzzy hits shown in Fig. 4.4, but here the query is much simpler. Intuitively, the query asks for any sequence that starts and ends with both knees stretched, regardless of what happens in between.

Not all fuzzy queries lead to sensible adaptive fuzzy hits. For example, the fuzzy query $\vec{V} = (V_0, V_1, V_2)$ with $V_0 = \{\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\}$, $V_1 = \{\begin{pmatrix} 1 \\ 0 \end{pmatrix}\}$, and $V_2 = \{\begin{pmatrix} 0 \\ 1 \end{pmatrix}\}$ yields no adaptive

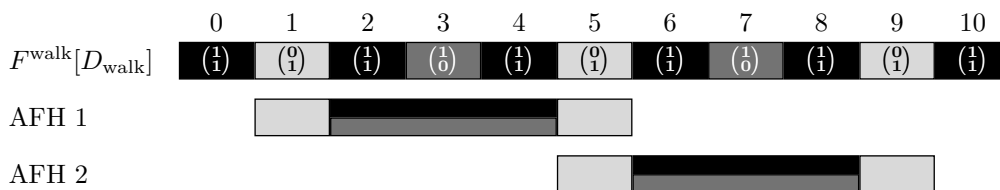


Figure 4.5. Upper row: feature sequence $\vec{w} = F^{\text{walk}}[D_{\text{walk}}]$. Below: two adaptive fuzzy hits (AFH) for $\vec{V}_{\text{walk},3}$.

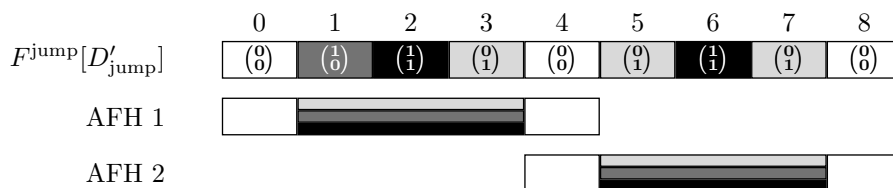


Figure 4.6. Upper row: feature sequence $F^{\text{jump}}[D'_{\text{jump}}]$. Below: two adaptive fuzzy hits (AFH) for $\vec{V}_{\text{jump},3}$ in $F^{\text{jump}}[D'_{\text{jump}}]$.

fuzzy hits on the document $D = (w_0, w_1, w_2) = ((\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}), (\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}), (\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}))$, even though the three feature vectors of the document appear as elements of the three corresponding fuzzy sets. This is due to the greedy strategy of choosing the maximal index $k_1 \geq 0$ with $w_m \in V_0$ for all $m = 0, \dots, k_1 - 1$; here, we obtain $k_1 = 2$, so an adaptive fuzzy hit for \vec{V} would now have to satisfy $w_2 \in V_1$, which is not the case. The underlying problem is that the fuzzy sets V_0 and V_1 have a nonempty intersection, which makes the transition between these two sets during the matching process inherently ill-defined. We therefore only allow *admissible* fuzzy queries with $V_n \cap V_{n+1} = \emptyset$ for $n = 0, \dots, N - 1$. In case a query is not admissible, one possible strategy is to scan the query (from 0 to N) for fuzzy sets V_n that are supersets of their neighbors V_{n-1} and V_{n+1} (where $V_{-1} = V_{N+1} := \emptyset$). Such V_n are then replaced by $V_n \setminus (V_{n-1} \cup V_{n+1})$, leading to an admissible query, see [Dem06].

Finally, we want to note that fuzzy search can be complemented by the concept of κ -mismatch search, see [Rib06, CK04]. Here, one introduces another degree of inexactness by permitting up to $\kappa < N$ of the fuzzy sets in a query $\vec{V} = (V_0, V_1, \dots, V_N)$ to completely disagree with the database sequence \vec{w} . To maintain a certain degree of control over this mismatch mechanism, it is possible to restrict the mismatchable fuzzy sets within \vec{V} .

4.2 Efficient Computation of Hits

Exact hits and fuzzy hits can be computed efficiently by standard indexing techniques based on inverted lists, see, e.g., [WMB99]. In this section, we first review the relevant data structures and algorithms for exact and fuzzy retrieval. Then, we introduce an extension that enables efficient computation of adaptive fuzzy hits by means of the same inverted list index.

4.2.1 Indexing based on Inverted Lists

To index our database \mathcal{D} with respect to the feature function F , we store an *inverted list* $L(v)$ for each feature vector $v \in \{0, 1\}^f$ consisting of the indices $k \in [0 : M]$ of the sequence $\vec{w} = (w_0, w_1, \dots, w_M)$ with $v = w_k$. The list $L(v)$ tells us which of the F -segments of D

exhibit the feature vector v . For our example $D = D_{\text{walk}}$ and $F = F^{\text{walk}}$, we obtain from (3.5) the inverted lists

$$\begin{aligned} L\left(\binom{1}{1}\right) &= \{0, 2, 4, 6, 8, 10\}, \\ L\left(\binom{0}{1}\right) &= \{1, 5, 9\}, \\ L\left(\binom{1}{0}\right) &= \{3, 7\}, \\ L\left(\binom{0}{0}\right) &= \emptyset. \end{aligned} \tag{4.6}$$

Inverted lists are sets represented as sorted, repetition-free sequences, accounting for efficient computations. Depending on the context, our notation for inverted lists and derived objects will switch between sequence and set notation. Note that consecutive entries of an inverted list differ by at least two since consecutive items in any F -feature sequence are distinct by construction.

In a preprocessing step, we construct an index I_F^D consisting of the 2^f inverted lists $L(v)$, $v \in \{0, 1\}^f$. Since the inverted lists store segment positions of the F -segmentation rather than frame positions, and since each segment position appears in exactly one inverted list, the index size is proportional to the number of segments of D , which is $M + 1$. Additionally, we store the segment lengths and/or boundaries so that the frame positions can be recovered. The time complexity of the indexing step is in $O(M)$ as well, assuming that the processing time for a single segment is constant. We refer to [CKK03, WMB99] for more details on indexing.

4.2.2 Computing Exact Hits and Fuzzy Hits

The set $H_{\mathcal{D}}(\vec{v})$ of all exact hits for \vec{v} in \mathcal{D} as defined in (4.2) can be evaluated efficiently by intersecting suitably shifted inverted lists:

$$\begin{aligned} H_{\mathcal{D}}(\vec{v}) &= \{k \in [0 : M] \mid (v_0 = w_k) \wedge (v_1 = w_{k+1}) \wedge \dots \wedge (v_N = w_{k+N})\} \\ &= \bigcap_{n \in [0 : N]} \{k \in [0 : M] \mid v_n = w_{k+n}\} \\ &= \bigcap_{n \in [0 : N]} (\{\ell \in [0 : M] \mid v_n = w_\ell\} - n) \\ &= \bigcap_{n \in [0 : N]} (L(v_n) - n), \end{aligned} \tag{4.7}$$

where the addition and subtraction of a list and a number is understood component-wise for every element in the list. The intersection in (4.7) can be computed iteratively, where we employ the following trick: instead of additively adjusting the input lists $L(v_n)$ (which are in general long) by $-n$, as suggested by (4.7), we adjust the lists appearing as intermediate results (which are in general much shorter) by $+1$ prior to each intersection step. One easily checks that after the final iteration, an adjustment of the resulting set by $-(N + 1)$ yields the set of exact hits. This idea gives rise to Algorithm 4.1, EXACTHITS.

Time complexity. The worst-case time complexity of this algorithm occurs when all successive lists that are to be intersected are identical, corresponding to the case where every element of an inverted list belongs to a hit. Then, the required number of operations is

Algorithm 4.1 EXACTHITS

Input: \vec{v} feature sequence of query.
 I_F^D index consisting of the 2^f inverted lists $L(v)$, $v \in \{0, 1\}^f$.
Output: $H_{\mathcal{D}}(\vec{v})$ list of exact hits.

Procedure:

- (1) $L^0 := L(v_0) + 1$
- (2) For $n = 0, \dots, N - 1$ compute $L^{n+1} := (L^n \cap L(v_{n+1})) + 1$
- (3) $H_{\mathcal{D}}(\vec{v}) = L^N - (N + 1)$

The loop invariant for step (2) is $L^n = H_{\mathcal{D}}((v_0, \dots, v_n)) + n + 1$ for $n = 0, \dots, N$.

proportional to the total number of elements in the inverted lists that are to be intersected, equalling the product of query length $(N + 1)$ and the number of hits.

In most practical cases, by far not all elements of the relevant inverted lists will have to be inspected. The time complexity is determined by the number of hits, the length distribution of the inverted lists, $L_0 := L(v_0), \dots, L_N := L(v_N)$, and by the complexity of the intersection operation on sorted lists. Assume w.l.o.g. that we are intersecting the two lists L_0 and L_1 , where $\lambda_0 := |L_0| \leq |L_1| =: \lambda_1$. Then, there are two possible intersection strategies: merging and binary search. The merging strategy performs a linear scan through both lists simultaneously, requiring $O(\lambda_0 + \lambda_1)$ operations. Merging is the method of choice in case $\lambda_0 \approx \lambda_1$. The binary search strategy, on the other hand, performs a linear scan through L_0 and looks up each element of L_0 in L_1 using binary search, adding the element to the result list in case the search is successful. This approach requires $O(\lambda_0 \log \lambda_1)$ operations and becomes faster than merging if $\lambda_0 \ll \lambda_1$.

Due to the commutativity of set intersection, the actual evaluation order of the intersection (4.7) is irrelevant. Also note that the resulting list of hits, $H_{\mathcal{D}}(\vec{v})$, is always a subset of the shortest list—in other words, the rarest feature vector determines the maximum number of hits. Therefore, a more efficient implementation could start with the shortest list and proceed in the order of ascending list lengths so as to cut down the size of the intermediary results as quickly as possible, see [CKK03]. However, for didactic reasons that will become apparent in the next section, we deliberately used a fixed processing sequence ($n = 0, \dots, N$) in Algorithm 4.1.

Example. As an illustration, we apply Algorithm 4.1 to our example $D = D_{\text{walk}}$, $F = F^{\text{walk}}$, and the query sequence $\vec{v}_{\text{walk},1} = \left(\binom{1}{0}, \binom{1}{1}, \binom{0}{1}\right)$, cf. Fig. 4.1. Recall from (4.6) that in this case $L\left(\binom{1}{1}\right) = \{0, 2, 4, 8, 10\}$, $L\left(\binom{0}{1}\right) = \{1, 5, 9\}$, and $L\left(\binom{1}{0}\right) = \{3, 7\}$.

- (1) $L^0 := L\left(\binom{1}{0}\right) + 1 = \{4, 8\}$
- (2) $L^1 := (\{4, 8\} \cap \{0, 2, 4, 8, 10\}) + 1 = \{5, 9\}$
 $L^2 := (\{5, 9\} \cap \{1, 5, 9\}) + 1 = \{6, 10\}$
- (3) $H_{\mathcal{D}}(\vec{v}_{\text{walk},1}) = L^2 - 3 = \{3, 7\}$

Fuzzy hits can be computed by the same algorithm with an additional preparation stage. Given a fuzzy query $\vec{V} = (V_0, V_1, \dots, V_N)$, we compute for each V_n , $n \in [0 : N]$, the sorted

Algorithm 4.2 ADAPTIVEFUZZYHITS

Input: \vec{V} fuzzy query, sequence of $N + 1$ fuzzy sets.
 $I_F^{\mathcal{D}}$ index consisting of the 2^f inverted lists $L(v)$, $v \in \{0, 1\}^f$.
Output: $H_{\mathcal{D}}^{\text{ad}}(\vec{V})$ list of adaptive fuzzy hits.

Procedure:

- (1) $R^0 := R(V_0) + T(V_0)$, $T^0 := T(V_0)$
- (2) For $n = 0, \dots, N - 1$ assume
 $R^n = (p_0, \dots, p_I)$, $T^n = (q_0, \dots, q_I)$,
 $R(V_{n+1}) = (r_0, \dots, r_J)$, $T(V_{n+1}) = (t_0, \dots, t_J)$,
 $R^n \cap R(V_{n+1}) = (p_{i_0}, \dots, p_{i_K}) = (r_{j_0}, \dots, r_{j_K})$, for suitable indices
 $0 \leq i_0 < \dots < i_K \leq I$ and $0 \leq j_0 < \dots < j_K \leq J$. Then define
 $R^{n+1} := (R^n \cap R(V_{n+1})) + (t_{j_0}, \dots, t_{j_K})$,
 $T^{n+1} := (q_{i_0}, \dots, q_{i_K}) + (t_{j_0}, \dots, t_{j_K})$.
- (3) $H_{\mathcal{D}}^{\text{ad}}(\vec{V}) = R^N - T^N$

The loop invariant for step (2) is that for $n = 0, \dots, N$, we have $R^n = H_{\mathcal{D}}^{\text{ad}}((V_0, \dots, V_n)) + T^n$, where T^n holds the number of segments for each hit in $H_{\mathcal{D}}^{\text{ad}}((V_0, \dots, V_n))$. In other words, the entries of R^n denote hit candidates and always point to the beginning of the next (potentially matching) group of segments.

union of inverted lists

$$L(V_n) := \bigcup_{v \in V_n} L(v). \quad (4.8)$$

Computing the union of the N sorted lists $L(v)$ can be done with $O(\log N \cdot \sum_v |L(v)|)$ operations using a parallel merge, where the logarithmic overhead stems from a min-heap data structure used to keep track of the current minimum. The lists $L(V_n)$ can then be used as the input to the exact hit algorithm, since the set $H_{\mathcal{D}}(\vec{V})$ can be expressed as

$$H_{\mathcal{D}}(\vec{V}) = \bigcap_{n \in [0:N]} (L(V_n) - n), \quad (4.9)$$

cf. (4.7) and (4.3). This formula also shows that the complexity of computing $H_{\mathcal{D}}(\vec{V})$ is proportional to $\sum_{n=0}^N |V_n|$ and not to $\prod_{n=0}^N |V_n|$, as could be suspected at first sight, cf. [CK04].

Example. As an illustration, we return to the example of Fig. 4.3, where the document D_{walk} is searched for the fuzzy query $\vec{V}_{\text{walk},2} = (V_0, V_1, V_2)$ with $V_0 = V_2 = \left\{ \binom{0}{1}, \binom{1}{0} \right\}$ and $V_1 = \left\{ \binom{1}{1} \right\}$. We have $L(V_0) = L(V_2) = \{1, 3, 5, 7, 9\}$ and $L(V_1) = \{0, 2, 4, 6, 8, 10\}$. Applying the algorithm, we obtain

- (1) $L^0 := L(V_0) + 1 = \{2, 4, 6, 8, 10\}$
- (2) $L^1 := (\{2, 4, 6, 8, 10\} \cap \{0, 2, 4, 6, 8, 10\}) + 1 = \{3, 5, 7, 9, 11\}$
 $L^2 := (\{3, 5, 7, 9, 11\} \cap \{1, 3, 5, 7, 9\}) + 1 = \{4, 6, 8, 10\}$
- (3) $H_{\mathcal{D}}(\vec{V}_{\text{walk},2}) = L^2 - 3 = \{1, 3, 5, 7\}$

4.2.3 Computing Adaptive Fuzzy Hits

It is possible to compute the set $H_{\mathcal{D}}^{\text{ad}}(\vec{V})$ efficiently using the same index $I_F^{\mathcal{D}}$ as for the case of exact hits, see also [MRC05c]. Before describing the algorithm, we need to introduce some more notation. Note that the list $L(V)$ for a fuzzy set V may contain sequences of consecutive segment indices, i. e., ascending runs $k_0, k_0 + 1, k_0 + 2, \dots$ (opposed to an inverted list $L(v)$). We consider sequences of consecutive segment indices of maximal length in $L(V)$. Suppose $L(V)$ consists of $K + 1$ such sequences with starting segments $r_0 < r_1 < \dots < r_K$ and lengths t_0, t_1, \dots, t_K ; then we define $R(V) := (r_0, r_1, \dots, r_K)$ and $T(V) := (t_0, t_1, \dots, t_K)$. For example, if $L(V) = (2, 4, 5, 6, 9, 10)$, then $R(V) = (2, 4, 9)$ and $T(V) = (1, 3, 2)$. Obviously, one can reconstruct $L(V)$ from $R(V)$ and $T(V)$. Note that by the maximality condition one has $r_k + t_k < r_{k+1}$ for $0 \leq k < K$. Extending the algorithm in Sect. 4.2.2, we can then compute the set $H_{\mathcal{D}}^{\text{ad}}(\vec{V})$ as described in Algorithm 4.2, ADAPTIVEFUZZYHITS.

Time complexity. The asymptotic complexity of Algorithm 4.2 is the same as that of Algorithm 4.1 since the overhead for handling the sequences T^n is proportional to the number of operations performed on the R^n .

Example. To illustrate the algorithm, we return to the example of Fig. 4.5. From the lists $L(V_0) = L(V_2) = (1, 5, 9)$ and $L(V_1) = (0, 2, 3, 4, 6, 7, 8, 10)$, we obtain $R(V_0) = R(V_2) = (1, 5, 9)$, $T(V_0) = T(V_2) = (1, 1, 1)$ and $R(V_1) = (0, 2, 6, 10)$, $T(V_1) = (1, 3, 3, 1)$. Then

- (1) $R^0 := (1, 5, 9) + (1, 1, 1) = (2, 6, 10)$
 $T^0 := (1, 1, 1)$
- (2) $R^1 := ((2, 6, 10) \cap (0, 2, 6, 10)) + (3, 3, 1) = (5, 9, 11)$
 $T^1 := (1, 1, 1) + (3, 3, 1) = (4, 4, 1)$
 $R^2 := ((5, 9, 11) \cap (1, 5, 9)) + (1, 1) = (6, 10)$
 $T^2 := (4, 4) + (1, 1) = (5, 5)$
- (3) $H_{\mathcal{D}}^{\text{ad}}(\vec{v}) = R^2 - T^2 = (1, 5)$.

4.3 A Retrieval System Using the Query-By-Example Paradigm

The indexing and retrieval techniques that have been introduced so far can be put to use in a variety of query modes, see also [DRME06a]. Here, the possibilities range from isolated pose-based queries (where a query consists of a single feature vector), over manually specified feature sequences or fuzzy sets, up to *query-by-example* (QBE). In this section, we will discuss the QBE mode, which is also summarized in Fig. 4.7 and Fig. 4.8.

Prior to the actual *preprocessing step*, an expert has to design a global feature function F that covers all possible query requirements and provides the user with an extensive set of semantically rich features. In other words, it is not imposed upon the user to specify such features (even though this is also possible). Having fixed a feature function F , an index $I_F^{\mathcal{D}}$ is constructed for a given database \mathcal{D} and stored on disk, see Sect. 4.2.1. As an example, one may use the feature set F of Tab. 3.2, which comprises $f = 39$ features. Note that this feature set has been specifically designed to focus on full-body motions. However, the described indexing and retrieval methods are generic, and the proposed test feature set may be replaced as appropriate for the respective application.

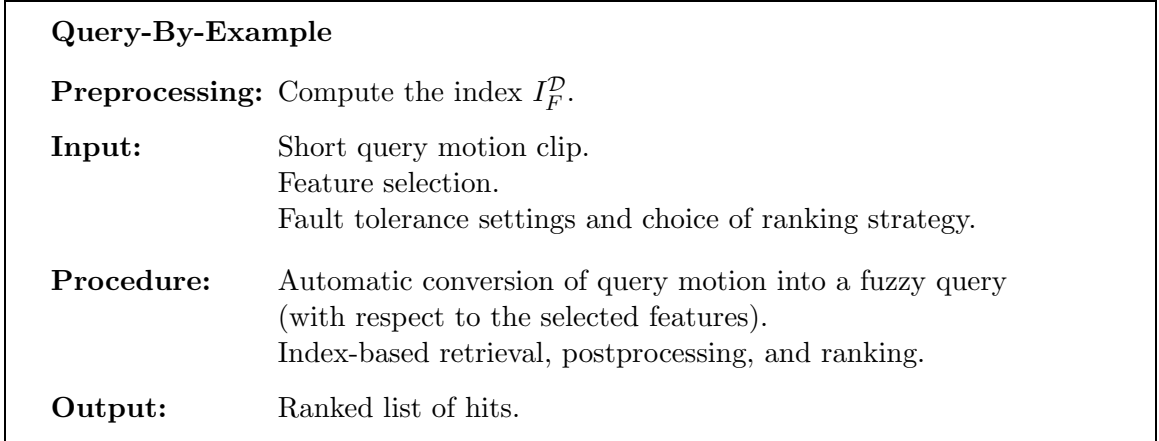


Figure 4.7. Overview of the retrieval process based on the query-by-example paradigm.

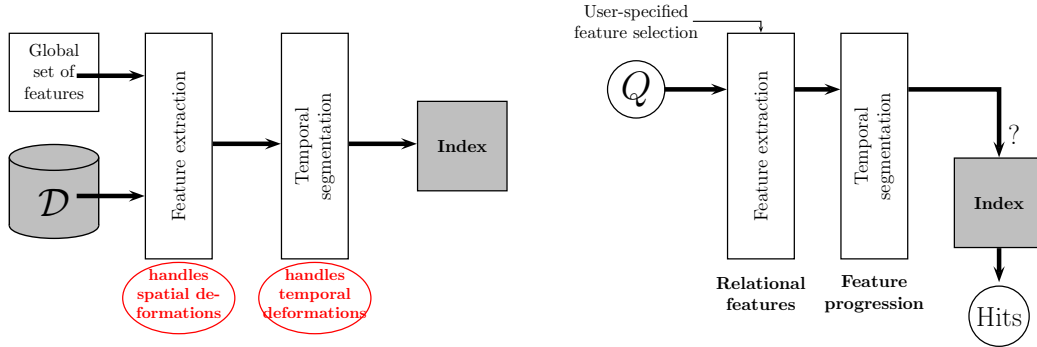


Figure 4.8. **Left:** The preprocessing stage. **Right:** The query stage.

We assume that the *user input* consists of a short query motion clip $Q : [1 : T] \rightarrow \mathcal{P}$. Furthermore, the user should be able to incorporate additional knowledge about the query, e.g., by selecting or masking out certain body areas in the query. This is important to account for partial similarity, see Sect. 2.4.2. For example, in movements such as “walking” or “punching” only the lower or the upper part of the body may be of interest. To this end, the user selects $f' \leq f$ relevant features from the given global feature set (i.e., components of F), where each feature expresses a certain relational aspect and refers to specific parts of the body. The query-dependent specification of motion aspects then determines the desired notion of similarity. In addition, parameters such as fault tolerance and the choice of a ranking or postprocessing strategy can be adjusted.

In the retrieval *procedure*, the query motion, Q , is translated into a feature sequence $\vec{v} = F[Q] := (v_0, \dots, v_N)$. The user-specified feature selection has to be encoded by a suitable fuzzy query, where the irrelevant features correspond to alternatives in the corresponding feature values. Denoting the coordinate functions of $F : \mathcal{P} \rightarrow \{0, 1\}^f$ as F^i , $i \in [1 : f]$, a user-specified feature selection can be encoded as an ordered subset $S := \{s_1, \dots, s_{f'}\} \subseteq [1 : f]$. We then denote the user-restricted feature function as $F^S := (F^{s_1}, \dots, F^{s_{f'}})$. Similarly, we denote the restriction of a feature vector $v \in \{0, 1\}^f$ to the components denoted by S as v^S . An exact query with respect to F^S can now be simulated as an adaptive fuzzy query with

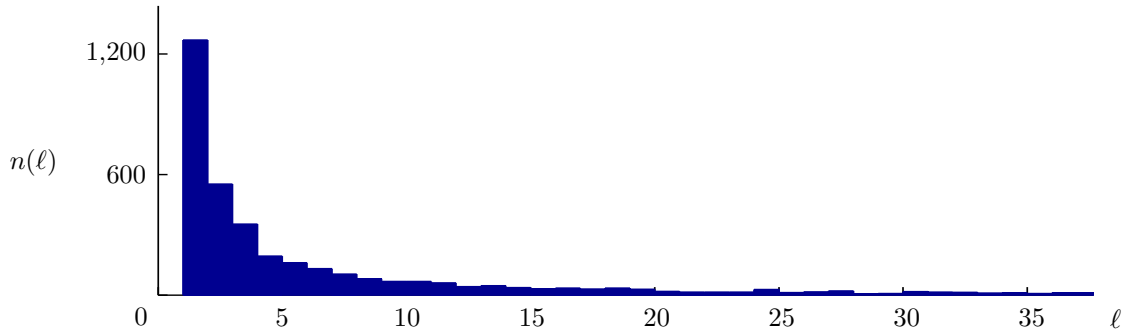


Figure 4.9. Typical distribution of list lengths for an index of 108,242 segments. The underlying feature function has $f = 14$ components, corresponding to a theoretical number of 16,384 inverted lists. Out of these, 13,008 lists are empty. The histogram shows the number of inverted lists, $n(\ell)$, that have length $0 < \ell \leq 38$.

respect to F , which allows us to perform all operations on the single index $I_F^{\mathcal{D}}$ —the major advantage being that we do not have to recompute the indexes $I_{FS}^{\mathcal{D}}$ for every user selection S . The appropriate adaptive fuzzy query, $\vec{V} = (V_0, \dots, V_N)$, is defined for $n \in [0 : N]$ via

$$V_n := \left\{ v \in \{0, 1\}^f \mid v^S = v_n^S \right\}. \quad (4.10)$$

Note in particular that \vec{V} is always an admissible fuzzy query, since adjacent elements of the underlying feature sequence, \vec{v} , are distinct.

In the next step, the adaptive fuzzy hits for \vec{V} are efficiently computed by the techniques described in Sect. 4.1.3. Then, the hits may be postprocessed to convert segment positions back to actual frame positions, using the supplementary data structures stored with the index, see Sect. 4.2.1. Finally, the resulting hits can be ranked as described in Sect. 4.3.1.

The attentive reader may have noticed a problem with the indexing step: in practice, the number 2^f of inverted lists is far too large. For example, in our case of $f = 39$, we obtain 2^{39} lists. Luckily, typical mocap databases such as the 210-minute database \mathcal{D}^{210} (see Sect. 5.1.2) only yield a limited number of different feature vectors, leading to a limited number of nonempty inverted lists. Fig. 4.9 shows a typical distribution of list lengths, exhibiting a very quick decay. The underlying feature function corresponds to a subset of F , comprising the 14 relational features marked as “u” (for “upper body”) in Tab. 3.2. There are only 3,376 out of a possible 16,384 inverted lists that are nonempty. Thus, a viable option to cope with the problem of a growing number of inverted lists with growing f would be to store and process only the nonempty inverted lists.

However, typical queries are posed with only very few selected features, say, 2–6 features. Converting an exact query with respect to 6 features into an adaptive fuzzy query with 39 features would lead to very large fuzzy sets containing $2^{39-6} = 2^{33}$ feature vectors. Even though only the feature vectors pertaining to nonempty lists would have to be considered, this approach entails an unnecessarily large number of union operations, see Eqn. 4.8. In other words, the problem is that indexing with respect to $f = 39$ features disassembles the database into pieces that are much too small for a typical query—hence, the adaptive fuzzy algorithm has to reassemble these pieces to achieve an adequate temporal granularity.

To avoid this computational overhead, we resort to the following approximation: we di-

vide the set of our 39 boolean features into the three sets F_ℓ (12 features), F_u (14 features), and F_m (13 features) as indicated by Table 3.2. The subscripts “ ℓ ”, “ u ”, and “ m ” stand for “lower body”, “upper body”, and “mix”, respectively. Identifying each feature set with the corresponding feature function, we then construct separate indexes $I_\ell^{\mathcal{D}}$, $I_u^{\mathcal{D}}$, and $I_m^{\mathcal{D}}$. Then, retrieval amounts to querying the individual indexes and postprocessing the resulting hits by additional merging and/or intersecting operations, see [DRME06a, Dem06]. However, the number of such additional operations is by far outweighed by the savings resulting from the significantly reduced overall number ($2^{12} + 2^{14} + 2^{13}$) of inverted lists. Also note that such a postprocessing of hits obtained from queries on different indices is not strictly equivalent to adaptive fuzzy retrieval on the full feature set, since the segmentations differ for each of the indexes. A minor drawback is that each of the three indexes $I_\ell^{\mathcal{D}}$, $I_u^{\mathcal{D}}$, and $I_m^{\mathcal{D}}$ requires an amount of memory linear in the overall number of F_ℓ -, F_u -, and F_m -segments in \mathcal{D} , respectively. This effect, however, is largely attenuated by the fact that segment lengths with respect to F_ℓ , F_u , and F_m are generally larger compared to F -segment lengths, resulting in fewer segments.

4.3.1 Ranking Strategies and Postprocessing of Hits

The feature function F is chosen so as to cover a broad spectrum of aspects appearing in all types of motions. Therefore, considering a specific motion class, many of the features are irrelevant for retrieval and should be masked out to avoid a large number of false negatives. Here, *false negatives* refer to database motion clips that are similar to the query but that are not retrieved by the system (e. g., due to over-specification). On the other hand, masking out a large number of features may lead to a larger percentage of *false positives*, i. e., hits that are not logically related to the query but coincidentally reveal the same relational aspects encoded by the user-specified features.

One important observation is that even though many useless hits might have been retrieved, one can easily eliminate most of the false positives in a postprocessing step. While the segment lengths (measured in frames) are left unconsidered in the retrieval process, they often encode characteristic timing information. Considering such timing information, the hits can be ranked by comparing the segment lengths of the respective hit with the corresponding segment lengths of the query motion. We assume a query of length $N + 1$, where the query segments have lengths ν_n frames for $n \in [0 : N]$. Furthermore, we think of a corresponding hit in which the matching segments have lengths μ_n , $n \in [0 : N]$. Then, we define the following ranking functions:

$$r^{\ell^1}(\nu_0, \mu_0, \dots, \nu_N, \mu_N) := \frac{1}{N+1} \sum_{n=0}^N |\nu_n - \mu_n| \quad (4.11)$$

$$r^{\log}(\nu_0, \mu_0, \dots, \nu_N, \mu_N) := \frac{1}{N+1} \sum_{n=0}^N \left| \log \frac{\nu_n}{\mu_n} \right|. \quad (4.12)$$

The first ranking function, r^{ℓ^1} , simply measures the average absolute deviation of corresponding segment lengths. In our experiments, this ranking function performed well if the query and logically related hits only differed by minor, local time scalings. The second ranking function, r^{\log} , is invariant to global time scalings since it only considers ratios of segment lengths (instead of absolute durations). Additionally, the logarithm and the absolute value ensure

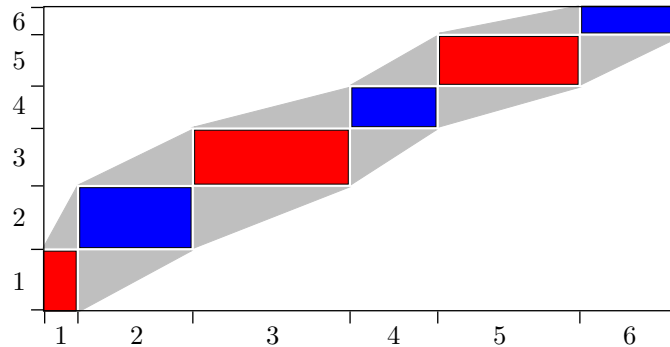


Figure 4.10. The segment-wise time alignment of the two walking motions shown in Fig. 3.8 (corresponding to the vertical axis) and Fig. 3.9 (b) (corresponding to the horizontal axis) can be refined via DTW-based methods restricted to the gray area.

that speeding up by a certain factor is penalized just as much as slowing down by the same factor. Using these ranking functions to sort the hits in ascending order according to their ranking value, many of the false positive hits (which often contain very short or extremely long matching segments) can be found at the end of the ranked list.

A further strategy is to postprocess the hits by means of more refined DTW-based methods using local distance measures as described in Sect. 3.5.2. Here, we propose a DTW-based ranking strategy that uses the Hamming distance between the binary vectors of the F -feature sequences. Depending on the user’s needs, the Hamming distance could only consider those features that were selected for the query or could also be based on all features of the global feature set. Let $c(n, m)$ denote the Hamming distance between the n^{th} vector in the F -feature sequence of the query and the m^{th} vector in the F -feature sequence of the hit. Then the hit’s ranking value is determined by the cost of the optimal warping path in the corresponding cost matrix $C = (c(n, m))$, see Sect. 3.5.2.

Here, the point is that even when working with a very coarse feature set and index-based retrieval (possibly leading to hundreds of hits), the data is still very efficiently reduced from a couple of hours (database) to a couple of minutes (hits), which is then well within reach of computationally expensive DTW-techniques. Furthermore, the cost matrices are typically very small because the query and the hits are compared at the segment level instead of the frame level, thus working on strongly downsampled motion representations. This ranking method is similar to the edit distance over the alphabet $\{0, 1\}^f$, see Sect. 2.7. Further details on DTW-based ranking can be found in [DRME06a, Dem06]

Finally, we sketch how index-based retrieval can be combined with DTW-based alignment techniques as introduced in Sect. 2.7. Recall that two motion clips are considered as similar if they possess (more or less) the same progression of relational features. Matching two such progressions obtained from similar motions can be regarded as a time alignment of the underlying motion data streams. Even though such alignments may be too coarse in view of applications such as morphing or blending, they are quite accurate with respect to the overall course of motion. We therefore suggest the following two-stage procedure: first use our index to efficiently compute a coarse, segment-based alignment. Then refine this alignment resorting to classical DTW-based techniques. The important point is that once a coarse alignment is known, the DTW step can be done very efficiently since the underlying cost matrix need only be computed within an area corresponding to the frames of the matched segments. This is also illustrated by Fig. 4.10, where the two walking motions of Fig. 3.8 and Fig. 3.9 (b)

Index	e	2^e	#(lists)	#(frames)	#(segs)	MB	$\frac{\text{bytes}}{\text{seg}}$	t_r	t_f	t_i	$\sum t$
I_ℓ^{60}	11	2048	409	425,294	21,108	0.72	35.8	26	10	6	42
I_ℓ^{180}	11	2048	550	1,288,846	41,587	1.41	35.5	71	26	13	110
I_u^{60}	12	4096	642	425,294	53,036	1.71	33.8	26	13	10	49
I_u^{180}	12	4096	877	1,288,846	135,742	4.33	33.4	71	33	25	129
I_m^{60}	8	256	55	425,294	19,067	0.60	33.0	26	20	3	49
I_m^{180}	8	256	75	1,288,846	55,526	1.80	34.0	71	54	12	137

Table 4.1. Feature computation and index construction. Running times are in seconds.

are aligned. In order to avoid alignment artifacts, the restricted area is slightly enlarged, as indicated by the gray area.

4.4 Experimental Results

We implemented our indexing and retrieval algorithms in Matlab 6.5 and tested them on a 3.6 GHz Pentium 4 with 1 GB of main memory. The test database was a subset of the Carnegie Mellon mocap database [CMU03], denoted as $\mathcal{D}_{180}^{\text{CMU}}$, containing more than one million frames of motion capture data (180 minutes sampled at 120 Hz). Since there is only a very coarse and heterogeneous ground truth annotation available for the CMU database, we resorted to evaluating representative queries. For these experiments, we used the feature function E comprising 31 relational features, see Tab. 3.1. E is divided into the three subsets E_ℓ , E_u , and E_m .

4.4.1 Indexing

We denote the indexes corresponding to E_ℓ , E_u , and E_m by I_ℓ^{180} , I_u^{180} , and I_m^{180} , respectively. In its columns, Table 4.1 shows the number e of feature components, the number 2^e of inverted lists, the number of nonempty inverted lists, the overall number of frames in the database, the overall number of segments of the corresponding segmentation, the index size in MB, the number of bytes per segment, and four running times t_r , t_f , t_i , and $\sum t$, measured in seconds. t_r is the portion of running time spent on data read-in, t_f is the feature extraction time, t_i is the inverted list build-up time, and $\sum t$ is the total running time. To demonstrate the scalability of our result, we quote analogous numbers for the indexes I_ℓ^{60} , I_u^{60} and I_m^{60} built from a subset $\mathcal{D}_{60}^{\text{CMU}}$ of $\mathcal{D}_{180}^{\text{CMU}}$ corresponding to 60 minutes of motion capture data. The total size of $\mathcal{D}_{180}^{\text{CMU}}$ represented in the text-based AMC motion capture file format (App. D) was 600 MB, a more compact binary double precision representation required about 370 MB. Typical index sizes ranged between 0.7 and 4.3 MB, documenting the drastic amount of data reduction our scheme achieves.

Table 4.1 shows that the number of segments (with respect to E_ℓ , E_u , and E_m) was only about 3 to 12 percent of the number of frames contained in the database. Observe that index sizes are proportional to the number of segments: the average number of bytes per segment is constant for all indexes. The total indexing time is *linear* in the number of frames. This fact is very well reflected by the table: for example, it took 42 seconds to build I_ℓ^{60} , which is roughly one third of the 110 seconds that were needed to build I_ℓ^{180} . Note that more than half of

Type, #(segs)	1–9 hits			10–99 hits			≥ 100 hits		
	μ_h	σ_h	μ_t (ms)	μ_h	σ_h	μ_t (ms)	μ_h	σ_h	μ_t (ms)
exact, $ Q = 5$	3.0	2.4	16	44	28	20	649	567	144
exact, $ Q = 10$	1.7	1.6	17	34	22	26	239	147	71
exact, $ Q = 20$	1.1	0.6	19	32	26	36	130	5	52
fuzzy, $ Q = 5$	3.6	2.5	23	44	27	29	1,878	1,101	291
fuzzy, $ Q = 10$	2.4	2.1	28	40	26	35	1,814	1,149	281
fuzzy, $ Q = 20$	2.0	1.9	42	36	24	35	1,908	1,152	294

Table 4.2. Statistics on 10,000 random queries in I_u^{180} using different query modes and query sizes, grouped by the number of hits, h . μ_h and σ_h are the average/standard deviation of h for the respective group, μ_t is the average query time in milliseconds.

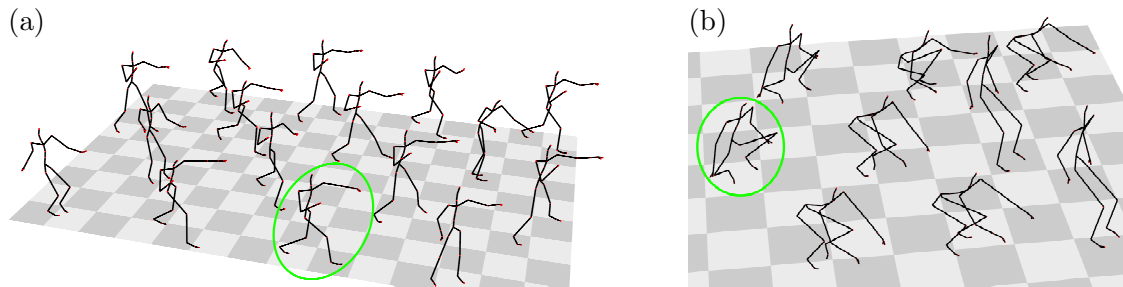


Figure 4.11. Selected frames from (a) 16 query-by-example hits for a left hand punch and from (b) 9 query-by-example hits for a squatting motion. The query clips are highlighted. Query features for (a): E_1, E_2, E_7, E_8 , query features for (b): $E_{17}, E_{18}, E_{19}, E_{22}, E_{23}$; see Table 3.1.

the total indexing time was spent on reading in the data, e.g., 71 seconds for the 180-minute index. The scalability of our algorithms' running time and memory requirements permits us to use much larger databases than those treated by Kovar and Gleicher [KG04], where the preprocessing step to build a match web is quadratic in the number of frames (leading, e.g., to a running time of roughly 3,000 seconds for a database containing only 37,000 frames), see also Sect. 4.6.

4.4.2 Retrieval

The running time to process a query very much depends on the size of the database, the query length (the number of segments), the user-specified fuzziness of the query, as well as the number of resulting hits. In an experiment, we posed 10,000 random queries (guaranteed to yield at least one hit) for each of six query scenarios to the index I_u^{180} , see Table 4.2. For example, finding all exact E_u -hits for a query consisting of 5/10/20 segments took on average 16–144/17–71/19–52 milliseconds, depending on the number of hits. Finding all adaptive fuzzy E_u -hits for a query consisting of 5/10/20 segments, where each fuzzy set of alternatives had a size of 64 elements, took on average 23–291/28–281/42–294 ms.

Fig. 4.11 (a) depicts all 16 hits resulting from a query for a punch (retrieval time: 12.5 ms), where only the four features E_1/E_2 (right/left hand in front) and E_7/E_8 (right/left elbow bent) were selected, see Table 3.1. These four features induce a segmentation of the query consisting of six segments, which suffice to grasp the gist of the punching motion. Further reducing the number of features by selecting only E_2 and E_8 induces a 4-segment query sequence and results in 264 hits, comprising various kinds of punch-like motions involving

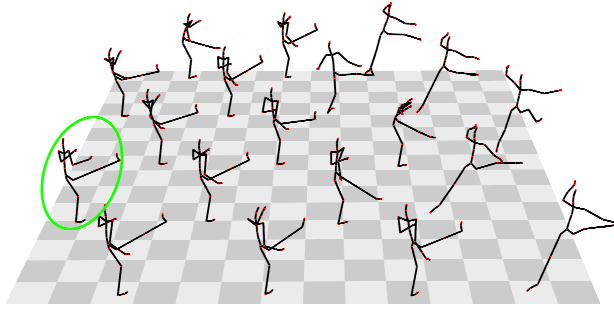


Figure 4.12. Selected frames from 19 query-by-example hits for a right foot kick. The query clip is highlighted. Query features: E_{15} , E_{16} , E_{19} , E_{20} ; see Table 3.1.

both arms. Finally, increasing the number of selected features by adding E_3/E_4 induces an 8-segment query sequence resulting in a single hit.

Fig. 4.11 (b) shows 9 hits out of the resulting 33 hits for a “squatting” motion (retrieval time: 18 ms) using the five features E_{17} , E_{18} , E_{19} , E_{22} , and E_{23} . The induced 5-segment query sequence is characteristic enough to retrieve 7 of the 11 “real” squatting motions contained in the database. Using a simple ranking strategy such as r^{ℓ^1} of Eqn. (4.11), these 7 hits appear as the top hits. The remaining 26 retrieved hits are false positives, two of which are shown on the right-hand side of Fig. 4.11 (b) as the skeletons “sitting down” on a virtual table edge. One reason for this type of false positives is that the relevant feature used in the query for the squatting motion thresholds the knee angle against a relatively high decision value of 120° . Hence, the knees of the sitting skeletons are just barely classified as “bent,” leading to the confusion with a squatting motion. Omitting the velocity features E_{17} and E_{18} again results in an induced 5-segment query, this time, however, yielding 63 hits (containing the previous 33 hits with the same top 7 hits). Among the additional hits, one now also finds jumping and sneaking motions.

Finally, Fig. 4.12 shows all 19 query results for a “kicking” motion (retrieval time: 5 ms) using E_{15} , E_{16} , E_{19} , and E_{20} . Out of these, 13 hits are actual martial arts kicks. The remaining six motions are ballet moves containing a kicking component. A manual inspection of $\mathcal{D}_{180}^{\text{CMU}}$ showed that there are no more than the 13 reported kicks in the database, demonstrating the high recall percentage our technique can achieve. Again, reducing the number of selected features leads to an increased number of hits. In general, a typical source of false positive hits is an inadequate choice of relational features in a query. For example, the ballet jumps in Fig. 4.12 were found as matches for a kicking motion because only the right leg was constrained by the query, leaving the left leg free to be stretched behind the body.

More retrieval results can be found at our web site [DRME06b]. Further results will also be presented in Sect. 5.2, where we compare index-based retrieval to retrieval using motion templates (MTs).

4.5 Discussion and Possible Extensions

In conclusion, our technique can efficiently retrieve high-quality hits with good precision/recall percentages provided that the user adequately selects a small number of features reflecting the important aspects of the query motion. This, in general, requires some experience as well as parameter tuning. However, most features have strong semantics, which makes feature

selection a very intuitive process. Also note that our algorithms were prototypically implemented in Matlab, hence an efficient C implementation can be expected to further speed up feature extraction, inverted list build-up, and querying by up to an order of magnitude.

Our retrieval method still suffers from a weakness that is associated with transitory segments as discussed in Sect. 4.1.1. Even though we showed in Sect. 4.1.2 how transitory segments can be absorbed by means of fuzzy queries, there are still some cases where the method fails. Returning to the “jumping” example of Sect. 4.1.2, recall that the constructed fuzzy query was capable of coping with transitions such as $\begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, or $\begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. However, the direct transition $\begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, where both features values switch at the same time, has still not been covered. This is due to the inability of our retrieval methods to express *insertions* and *deletions* within a sequence: the fuzzy set handling the transitory segment *must* be matched and may not be omitted in case it is not needed. Recall that the less efficient technique of DTW can handle such insertions and deletions: at this point, the tradeoff between efficiency and retrieval quality, corresponding to the choice between index-based retrieval and DTW, becomes apparent. To remedy the problem of simultaneous changes of feature values (which occurs very rarely at a sampling rate of 120 Hz), we can modify our retrieval algorithm so as to handle “dummy segments” within queries for such transitions, thereby largely increasing the time complexity, see also [DRME06a].

In view of fully automatic motion retrieval, as required for motion reuse techniques such as [AFO03, CH05, PB02], the required user interaction for manual feature selection turns out to be a drawback. Even though the manual specification enables the user to incorporate prior knowledge about the query, thus providing a great deal of flexibility, an option for fully automatic feature selection would be important as well.

The following hierarchical approach is a possible ad-hoc solution to this problem. We exemplarily discuss the idea by means of the feature function E_ℓ , which consists of 11 features.

- (0) Retrieve all hits with respect to E_ℓ .
- (1) If there are not enough hits, mask out one feature. This can be done in $\binom{9}{1} = 9$ ways, leading to feature sets E^1, \dots, E^9 . Then, process the query with respect to each of these feature sets and take the union of all retrieved hits (here, one has to eliminate multiplicities in the hits).
- (2) If there are still not enough hits, we mask out any two features. This can be done in $\binom{9}{2} = 36$ ways, leading to features sets $E^{12}, E^{13}, \dots, E^{19}, E^{23}, \dots, E^{89}$. In analogy to step (2), we process the query with respect to each of these feature sets.
- (3) Continue in an analogous fashion until there are enough hits.

All retrieval operations can be performed on the single index I_u . Furthermore, note that the hits retrieved in (0) are contained in the hits retrieved in (1), which in turn are contained in the hits retrieved in (2), and so on. This induces a retrieval hierarchy, where the number of hits increases but, in general, the quality of the hits decreases. In particular, masking out features that are specific for the query may lead to a large number of unrelated and useless hits. As an example, our “kicking” query of Fig. 4.12 yielded 1 hit in step (0), 9 hits in step (1), 15 hits in step (2), and 251 hits in step (3). The quality of the hits in step (0)–(2) was still acceptable, while step (3) led to many meaningless hits. Recall, however, that these results can be refined by the ranking strategies of Sect. 4.3.1.

A more sophisticated approach to automatic feature selection will be presented in Chap. 5. Based on relational features and suitable training motions, we will derive a class representation in form of a so-called motion template, which encodes the characteristic and the variational aspects of the motion class. Such motion templates can be used to locally and globally adjust the feature selection by masking out the variational aspects associated with the respective motion class. This method facilitates fully automatic motion retrieval and classification on large motion databases.

4.6 Related Work

In view of massively growing multimedia databases of various types and formats, efficient methods for indexing and content-based retrieval have become an important issue. Vast literature exists on indexing and retrieval in text, image, and video data, see, e. g., Witten et al. and Sebe et al. [WMB99, SLZ⁺03] and references therein. For the music scenario, Clausen and Kurth [CK04] give a unified approach to content-based retrieval; their group theoretical concepts are a generalized form of our efficient retrieval technique. However, their framework does not consider adaptive fuzzy search. The problem of indexing large time series databases has also attracted great interest in the database community, where DTW-based and LCS-based retrieval approaches are pursued, see, e. g., the work of Last et al., Keogh, and Vlachos et al. [LKB04, Keo02, VHKG06].

Due to possible spatio-temporal variations, the difficult task of identifying similar motion segments still bears open problems. Many of the previous approaches to motion comparison are based on DTW in conjunction with features that are semantically close to the raw data, using 3D positions, 3D point clouds, joint angle representations, or PCA-reduced versions thereof, see [CVKG03, FF05, HPP05, KG04, KPZ⁺04, PB02, SKK04, WCYL03, YSLG05]. We exemplarily discuss some of these approaches.

Pullen and Bregler [PB02, Pul02]. Recall from Sect. 2.6 the semi-local similarity measure that is employed by Pullen and Bregler in the context texturing and synthesis of motion capture data. The input to their system are rough, keyframed motion sketches, such as an animator's sketch of the lower body motion in a walking sequence. Exploiting the fact that different joint angles are typically highly correlated, the system tries to use a given angle trajectory such as the hip angle to infer the missing upper-body angles in the walking sequence. Based on this idea, similar mocap clips are retrieved from an underlying mocap database using certain angle trajectories as the query. For full synthesis of certain body parts' angle trajectories, the retrieval results are then suitably stitched together to form a smooth and visually appealing animation. Texturing uses the same principle, but only certain high- and mid-frequency bands resulting from an MRA analysis [BW95] of the data are modified.

The retrieval works as follows. First, the given query angle trajectories and the mocap database are analyzed by MRA, and only a suitable low-frequency band is retained for the following steps. Here, Pullen and Bregler [PB02, Pul02] make the assumption that the *motion content* is contained mainly in that low-frequency band, see also Sect. 2.3. Next, all trajectories are segmented according to the characteristics of a user-specified angle trajectory, for example the hip angle: local maxima and local minima of the hip trajectory are determined, and all trajectories are simultaneously segmented at the corresponding points in time. Building on the assumption that the chosen low-frequency band contains the motion content, one

may further assume that logically similar motions will yield similar segmentations due to a similar structure regarding local maxima and minima. Then, all query segments are processed sequentially, where local temporal deformations are accounted for by linearly stretching or compressing all segments from the mocap database (up to a factor of 4) to match the length of the respective query segment. Finally, the squared Euclidean distance between the query segment and the resampled database segments provides a ranking.

Liu et al. [LZWM05]. One problem of numerical features is their sensitivity towards pose deformations, as may occur in logically related motions. To achieve more robustness, Liu et al. use PCA and k -means clustering to transform motions described by their 3D trajectories into sequences of cluster centroids, which absorb spatio-temporal variations. This strategy is similar to our concept of temporal segmentation, but differs in the fact that their features have no associated semantics, while relational features do. Similar to our segment-based similarity measure and to the similarity measure of Pullen and Bregler [PB02], the technique by Liu et al. can be classified as semi-local (see Sect. 2.6), where the basic units of comparison are motion-dependent temporal segments. However, they provide no details on their retrieval method except that exact string matching as well as approximate string matching (a variant of DTW) have been successfully applied to compare a query's cluster transition signature against that of a large subset of the CMU mocap database [CMU03].

Forbes and Fiume [FF05]. Based on a quaternion representation, Forbes and Fiume learn a PCA space from range of motion data so as to accurately represent as many different poses as possible. To search for mocap clips, they project their motion database into the PCA space and apply the same projection to a given query clip, resulting in high-dimensional, time-dependent trajectories, see also Sect. 3.3.4 and Fig. 3.20 for the principle of PCA. Forbes and Fiume claim that the Euclidean distance in the PCA space is a good measure of similarity for individual poses. Under this assumption, they pursue the strategy of identifying motions by means of certain characteristic poses, which are defined as the poses that are most distant from the origin of the PCA space. See also [ACCO05] for alternative approaches to selecting characteristic poses. Given the characteristic pose of the query motion, Forbes and Fiume apply a spatial indexing data structure to find those database poses in the PCA space that are closest to the characteristic pose. After clustering the resulting database poses, the results are compared to the query's PCA trajectory using DTW and ranked according to their DTW distance. Some results for mocap databases with lengths ranging from 70 seconds to 11 minutes (120 Hz sampling rate) are reported, where query times are on the order of half a second.

Wu et al. [WCYL03]. Similar to Forbes and Fiume [FF05], Wu et al. proceed in two stages: they first identify start and end frames of possible candidate clips utilizing a pose-based index structure and then compute the actual distance from the query via DTW on SOM-based trajectory clusters.

Yu et al. [YSLG05]. A further attempt at designing a semi-local similarity measure is described by Yu et al. [YSLG05]. They propose a DTW-based motion retrieval system using *Labanotation* [Gue04, vL95], a versatile movement notation language developed in the dance community. Queries can either be specified by directly entering a Laban sequence, or by

providing a motion example. In the latter case, the query sequence is first converted to Labanotion using a method by Hachimura and Nakamura [HN01]. The motion database is also assumed to be annotated by a Laban transcription, and matching between the query and the database is performed via DTW. However, querying a 19,000-frame mocap database for a short motion clip comprising 150 frames takes 139 seconds using their method.

Match Web: Kovar and Gleicher [KG04]. In the approach of Kovar and Gleicher [KG04], numerically similar motions are identified by means of a DTW-based index structure termed *match web*. Recall that the key idea of DTW-based alignment between two motions $D : [1 : N] \rightarrow \mathcal{P}$ and $D' : [1 : M] \rightarrow \mathcal{P}$ was to compute a cost matrix C with respect to a local cost measure and then to derive a cost minimizing warping path, which typically runs through a “valley” of low cost in a neighborhood of the main diagonal of C . As was already mentioned in Sect. 2.7, the cost matrix C reveals further similarity relations between subsegments of D and D' , which are encoded by cost valleys running along secondary diagonals. This observation can be exploited for content-based motion retrieval based on self-similarity. Instead of comparing two different mocap data streams, the mocap data stream D is now compared to itself. Kovar and Gleicher use the 3D point cloud distance c^{3D} as the local similarity measure (Sect. 2.5.2), yielding the quadratic *self-similarity matrix* $C^{3D} \in \mathbb{R}^{N \times N}$. Fig. 4.13 shows an example of a self-similarity matrix for a gymnastics motion. Obviously, the optimal warping path within C^{3D} is just the main diagonal, which has a total cost of zero since the distance measure c^{3D} is definite (identical poses have zero distance). Besides the main diagonal, there are secondary diagonal paths of low costs. These paths denote segments of D that are similar to other segments of D .

This principle can be exploited to identify similar motion fragments contained in D , which can be thought of as the concatenation of many database motions, where we keep track of document boundaries in a supplemental data structure. Then, given a subclip of D that acts as a query motion clip Q , the task is to identify all motion subclips in D that are similar to Q . Such motion subclips are then referred to as *hits*. As an example, suppose the query consists of the first squat ($D(108:132)$) of the gymnastics motion of Fig. 4.13. We consider the part of the self-similarity matrix consisting of the “vertical stripe” above the query fragment, see Fig. 4.14 (a). In the next step, all diagonal paths of low cost are identified within this stripe. Finally, the projection of each such path onto the vertical axis corresponds to a hit, see Fig. 4.14 (b). In our example, there are two partial warping paths p^1 and p^2 of low cost running from cell (108, 108) to (132:132) and from cell (108:133) to (132:159), respectively. The projections of these paths result in the two hits $D(108 : 132)$ (the query itself) and $D(133:159)$ (the second squat).

Based on this general strategy, Kovar and Gleicher have developed an index structure named *match web*, which consists of all paths of low cost in a self-similarity matrix. Additionally the match web contains *bridges* of low total cost that connect certain paths if their endpoints are close to each other. The match web allows for fast retrieval of numerically similar motions. Kovar and Gleicher claim that a further step is possible: using a multi-step search spawning new queries from previously retrieved motions allows for the identification of logically similar motions using numerically similar motions as intermediaries. The authors report good retrieval results, which are particularly suitable for their blending application.

There are several problems associated with a retrieval strategy based on a (precomputed) self-similarity matrix as described above. First, being quadratic in the number of frames,

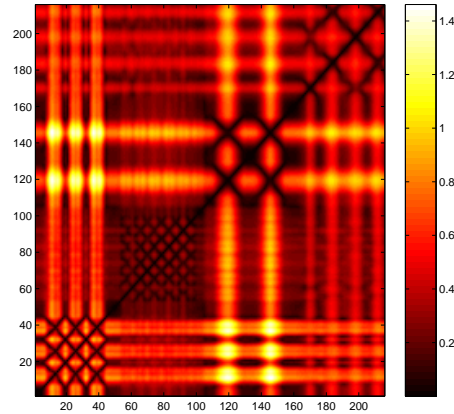


Figure 4.13. Self-similarity matrix for a database motion D (sampling rate 12 Hz) consisting of four jumping jacks ($D(1:41)$), a running motion ($D(42:107)$), two squats ($D(108:159)$), and two repetitions of an alternating elbow-to-knee motion ($D(160:215)$).

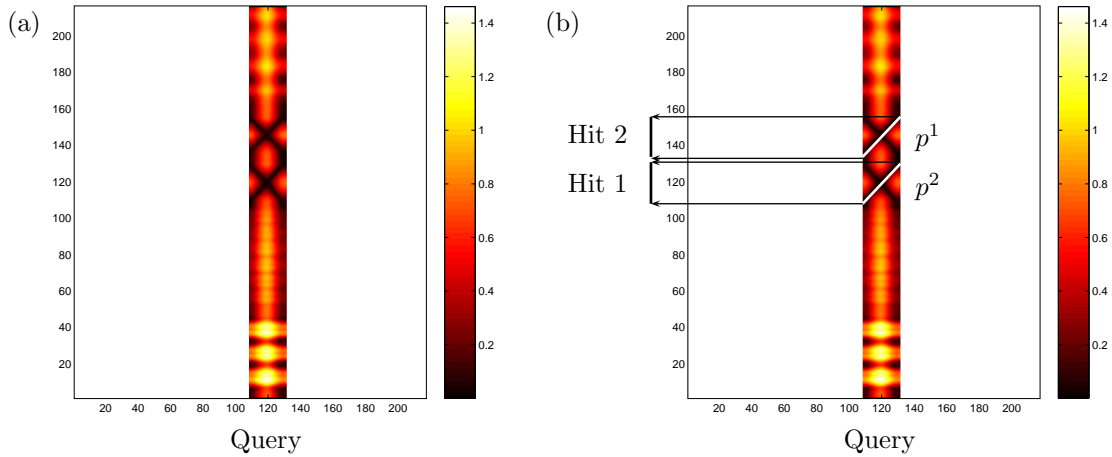


Figure 4.14. (a) Part of the self-similarity matrix shown in Fig. 4.13 consisting of the “vertical stripe” above the query motion $D(108:132)$ (corresponding to the first squat). (b) The two diagonal paths of low cost (white lines) correspond to the two squats contained in the database.

computing and storing the self-similarity matrix is computationally expensive. Hence, such a technique does not scale to large databases. Second, the suggested retrieval approach does not allow the user to incorporate a-priori knowledge of the motion. For example, if the user knows that the motion aspects of interest only regard the lower part of the body, he should be able to mask out irrelevant motion aspects such as the arm movements. Finally, the fact that the query motion must be contained in the database is a major drawback.

Chapter 5

Motion Templates for Retrieval and Classification

Based on [MR06b], we introduce in this chapter a generalization of relational feature matrices (Sect. 3.5.1): the concept of *motion templates* (MTs) is suited to express the essence of an entire class of motions. A motion template of *dimension* f and *length* K is a real-valued matrix $X \in [0, 1]^{f \times K}$. Each row of an MT corresponds to one relational feature, and time (in frames) runs along the columns, see Fig. 5.2 for an example. In the following, we assume that all MTs under consideration have the same fixed dimension f . Intuitively, an MT can be thought of as a “fuzzified” version of a feature matrix; for the proper interpretation of the matrix entries, we refer to Sect. 5.1, where we describe how to learn an MT from training motions by a combination of time warping and averaging operations. In Sect. 5.2.1, we then explain how MTs can be applied for fully automatic, efficient, and robust motion retrieval and classification. Additional results including a large number of illustrative videos can be found on our web site [MR06a].

5.1 Learning MTs from Example Motions

Initialization. Given a set of $\gamma > 0$ example motion clips for a specific motion class, such as the four cartwheels shown in Fig. 5.1 (left), our goal is to automatically learn a meaningful MT that grasps the essence of the class. We start by computing the feature matrices with respect to the feature set F of Tab. 3.2. The results are shown in Fig. 5.1 (right), where, for the sake of clarity, we only display a subset comprising ten features contained in F . From this point forward, we will consider feature matrices as a special case of MTs.

Weight vectors $\alpha \in \mathbb{R}_{>0}^K$ are attached to each of the MTs and initialized to $\alpha(k) = 1$ for all k . We then say that the k^{th} column $X(k)$ of X has weight $\alpha(k)$. During the learning procedure, the total weight $\bar{\alpha} := \sum_{k=1}^K \alpha(k)$ will always be at least one. These weights are used to keep track of the time warping operations: initially, each column of an MT corresponds to the real time duration of one frame, which we express by setting all weights $\alpha(k)$ to one. Subsequent time warping may change the amount of time that is allotted to an MT column. The respective weights are then modified so as to reflect the new time duration. Hence, the weights allow us to unwarped an MT back to real time, similar to the strategy used in [HPP05].

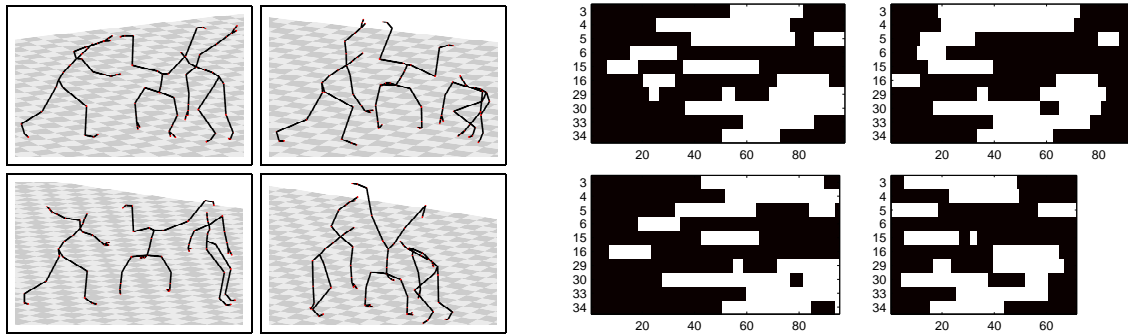


Figure 5.1. **Left:** Selected frames from four different cartwheel motions. **Right:** Corresponding relational feature matrices for selected features. The columns represent time in frames, whereas the rows correspond to boolean features encoded as black (0) and white (1). They are numbered in accordance with the features defined in Table 3.2.

Warping. Now, the goal is to compute a semantically meaningful average over the γ input MTs, which would simply be the arithmetic mean of the feature matrices if all of the motions agreed in length and temporal structure. However, our MTs typically differ in length and reflect the temporal variations that were present in the original motions. This fact necessitates some kind of temporal alignment prior to averaging. We do this by choosing one of the input MTs as the *reference MT*, R , and applying DTW (Sect. 2.7) to compute optimal alignments of the remaining MTs with R (we measure local distances of feature vectors by the Manhattan (ℓ^1) distance, which coincides with the Hamming distance for boolean feature vectors.) According to these optimal alignments, the MTs are locally stretched and contracted, where time stretching is simulated by duplicating MT columns, while time contractions are resolved by forming a weighted average of the columns in question. As indicated above, the weights α associated with an MT X must now be adapted accordingly: in case a column $X(\ell)$ was matched to n columns $R(k), \dots, R(k+n-1)$ of the reference, the new weights $\alpha'(k+i)$ are set to $\frac{1}{n}\alpha(\ell)$ for $i = 0, \dots, n-1$, i.e., the weight $\alpha(\ell)$ is equally distributed among the n matching columns. In case column $R(k)$ of the reference was matched to multiple columns of X , the new weight $\alpha'(k)$ is the sum of the weights of the matching columns in X .

Averaging. Now that all MTs and associated weight vectors have the same length as the reference MT, we compute the weighted average over all MTs in a column-wise fashion as well as the arithmetic mean $\bar{\alpha}$ of all weight vectors. Note that the total weight, $\bar{\alpha}$, equals the average length of the input motions. Fig. 5.2 (a) shows the results for our cartwheel example, where the top left MT in Fig. 5.1 acted as the reference. Finally, we unwarped the average MT according to the weight vector: column ranges with $\alpha(k) < 1$ are unwarped by contracting the respective MT columns into one average column (e.g., $k = 6, \dots, 10$ in Fig. 5.2 (a)), while columns with $\alpha(k) > 1$ are unwarped by duplicating the respective column (e.g., $k = 42$). Since in general, columns will not have integer or reciprocal integer weights, we additionally perform suitable partial averaging between adjacent columns such that all weights but the last are one in the resulting unwarped MT, see Fig. 5.2 (b). Note that the total weight, $\bar{\alpha}$, is preserved by the unwarping procedure. The average MT now constitutes a combined representation of all the input motions, but it is still biased by the influence of the reference MT, to which all of the other MTs have been aligned. Our experiments show that it

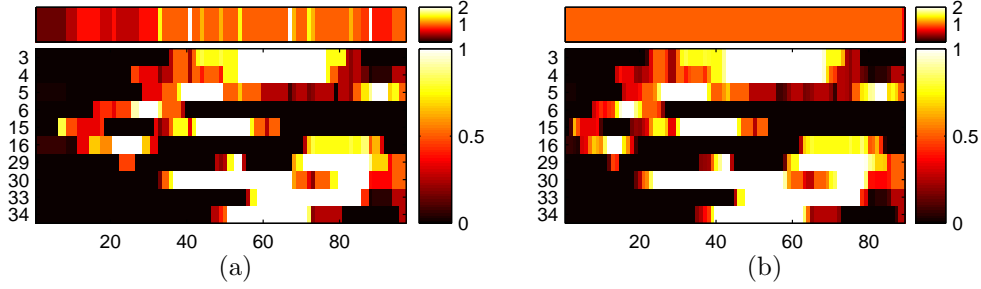


Figure 5.2. (a) Average MT and average weights computed from the MTs in Fig. 5.1 after all four MTs have been aligned with the top left MT, which acted as the reference. The MTs are coded in white (1), black (0), and shades of red and yellow for intermediary values. (b) Unwarped version of the MT in (a).

Motion Class \mathcal{C}	Comment	Size	γ	$\bar{\alpha}$	M	$t(\mathcal{C})$
CartwheelLeft	left hand first on floor	21	11	105.3	6	17.0
ElbowToKnee	start: rellow/lknee	27	14	36.6	5	4.9
JumpingJack	1 repetition	52	26	35.5	6	19.3
KickFrontRFoot	1 kick	30	15	53.3	5	9.4
KickSideRFoot	1 kick	30	15	48.9	6	10.1
LieDownFloor	start: standing pose	20	10	165.0	5	25.6
RotateRArmBwd3	3 times	16	8	80.8	4	3.8
RotateRArmFwd3	3 times	16	8	83.6	4	3.9
Squat	1 repetition	52	26	47.3	5	24.6
WalkSideRight3	3 steps	16	8	123.0	3	5.5

Table 5.1. \mathcal{D}^{MC} contains 10 to 50 different variations for each of its 64 motion classes. This table shows ten of the motion classes, along with their respective size, the size γ of the training subset, the average length $\bar{\alpha}$ in frames, as well as the number M of iterations and the running time $t(\mathcal{C})$ in seconds required to compute $X_{\mathcal{C}}$.

is possible to eliminate this bias by the following strategy: we let each of the original MTs act as the reference and perform for each reference the entire averaging and unwarping procedure as described above. This yields γ averaged MTs corresponding to the different references. Then, we use these γ MTs as the input to a second pass of mutual warping, averaging, and unwarping, and so on. The procedure is iterated until no major changes occur. Fig. 5.3 shows the output for $N = 11$ training motions.

5.1.1 Interpretation of MTs

An MT learned from training motions belonging to a specific motion class \mathcal{C} is referred to as the *class template* $X_{\mathcal{C}}$ for \mathcal{C} . Note that the weight vector does not play a role any longer. Black/white regions in a class MT, see Fig. 5.3, indicate periods in time (horizontal axis) where certain features (vertical axis) consistently assume the same values zero/one in all training motions, respectively. By contrast, red to yellow regions indicate inconsistencies mainly resulting from variations in the training motions (and partly from inappropriate alignments). Some illustrative examples will be discussed in Sect. 5.1.3. Intuitively, each row of such a class template can be viewed as an estimate of the temporal distribution of the corresponding feature’s values, which is highly class-dependent and thus an essential characteristic for \mathcal{C} . The

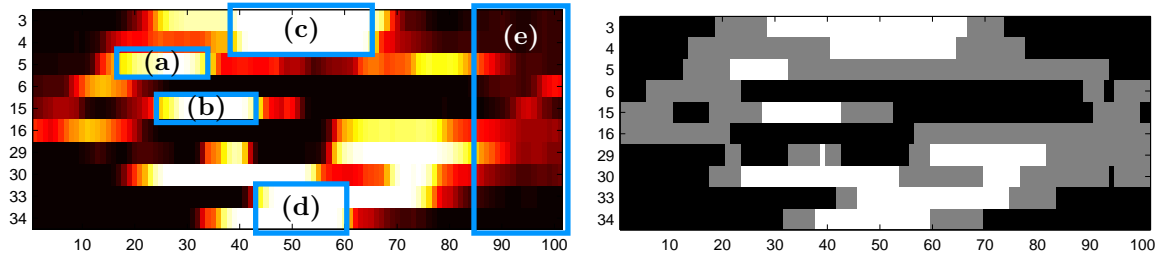


Figure 5.3. **Left:** Class MT for ‘CartwheelLeft’ based on $\gamma = 11$ training motions. The framed regions are discussed in Section 5.1.3. **Right:** Corresponding quantized class MT, see Sect. 5.2.1.

interpretation of a single matrix entry $x_{ik} := X_C(k)_i \in [0, 1]$ is that on average, a proportion of x_{ik} of the training motions assumed the feature value one for feature i at frame number k . Here, the semantics of a frame number k along a class template’s time axis is that of an average time position, which makes sense because corresponding columns of the MTs have been aligned.

5.1.2 Experimental Results

For our experiments, we systematically recorded several hours of motion capture data at a sampling rate of 120 Hz, containing a number of well-specified motion sequences, which were executed several times and performed by five different actors. Using this data, we built up the database \mathcal{D}_{210} consisting of roughly 210 minutes of motion data. Then, from \mathcal{D}_{210} , we manually cut out suitable motion clips and arranged them into 64 different classes. Each such motion class (MC) contains 10 to 50 different realizations of the same type of motion, covering a broad spectrum of semantically meaningful variations. For example, the motion class ‘CartwheelLeft’ contains 21 variations of a cartwheel motion, all starting with the left hand. The resulting *motion class database* \mathcal{D}^{MC} contains 1,457 motion clips, amounting to 50 minutes of motion data.

Table 5.1 gives an overview of some of the motion classes contained in \mathcal{D}^{MC} . We split up \mathcal{D}^{MC} into two disjoint databases \mathcal{D}^{MCT} and \mathcal{D}^{MCE} , each consisting of roughly half the motions of each motion class. The database \mathcal{D}^{MCT} served as the *training database* to derive the motion templates, whereas \mathcal{D}^{MCE} was used as a training-independent *evaluation database*. All databases were preprocessed by computing and storing the feature matrices. Here, we used a sampling rate of 30 Hz, which turned out to be sufficient in view of MT quality. The duration of the training motion clips ranged from half a second up to ten seconds, leading to MT lengths between 15 and 300. The number of training motions used for each class ranged from 7 to 26. Using 3 to 7 iterations, it took on average 7.5 s to compute a class MT on a 3.6 GHz Pentium 4 with 1 GB of main memory, see Table 5.1. For example, for the class ‘RotateRArmFwd3’, the total computation time was $t(\mathcal{C}) = 3.9$ s with $\bar{\alpha} = 83.6$, $N = 8$, and $M = 4$, whereas for the class ‘CartwheelLeft’, it took $t(\mathcal{C}) = 17.0$ s with $\bar{\alpha} = 105.3$, $N = 11$, and $M = 6$.

5.1.3 Examples

To illustrate the power of the MT concept, which grasps the essence of a specific type of motion even in the presence of large variations, we discuss the class template for ‘CartwheelLeft’ as

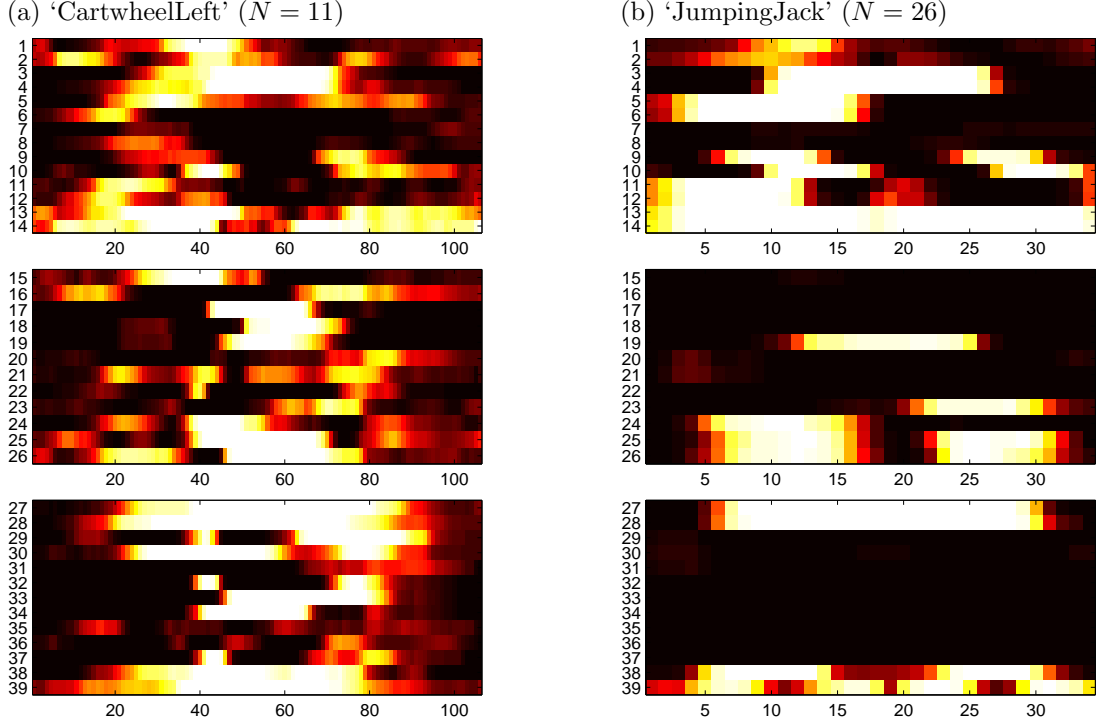


Figure 5.4. Class MT for (a) ‘CartwheelLeft’ and (b) ‘JumpingJack’ based on $N = 1$ and $N = 26$ training motions, respectively. For the sake of clarity, the rows corresponding to the 39 features are grouped according to the lower, upper, and mixed feature set, see Table 3.2.

a representative example. Fig. 5.3 shows the cartwheel MT learned from $N = 11$ example motions, which form a superset of the motions shown in Fig. 5.1. Recall that black/white regions in a class MT correspond to consistent aspects of the training motions, while colored regions correspond to variable aspects. The following observations illustrate that the essence of the cartwheel motion is captured by our class MT. Considering the regions marked by boxes in Fig. 5.3, the white region (a) reflects that during the initial phase of a cartwheel, the right hand moves to the top (feature F_5 in Table 3.2). Furthermore, region (b) shows that the right foot moves behind the left leg (F_{15}). This can also be observed in the first poses of Fig. 5.1. Then, both hands are above the shoulders (F_3, F_4), as indicated by region (c), and the actor’s body is upside down (F_{33}, F_{34}), see region (d) and the second poses in Fig. 5.1. The landing phase, encoded in region (e), exhibits large variations between different realizations, leading to the colored regions. Note that some actors lost their balance in this phase, resulting in rather chaotic movements, compare the third poses in Fig. 5.1. The complete class MT for ‘CartwheelLeft’ is shown in Fig. 5.4 (a).

Next, we study the relatively homogeneous class ‘JumpingJack’, which contains motions comprising one jumping cycle, starting and ending in the neutral standing pose. The homogeneity of the training motions is reflected by the absence of any major colored regions in the resulting class template (Fig. 5.4 (b)), which was derived from 26 different training motions. For the sake of clarity, we display the MTs as three separate parts corresponding to the features of the upper, lower, and mixed feature sets, see Table 3.2. The consistently black and white regions of the MT can be interpreted as follows: in all training motions,

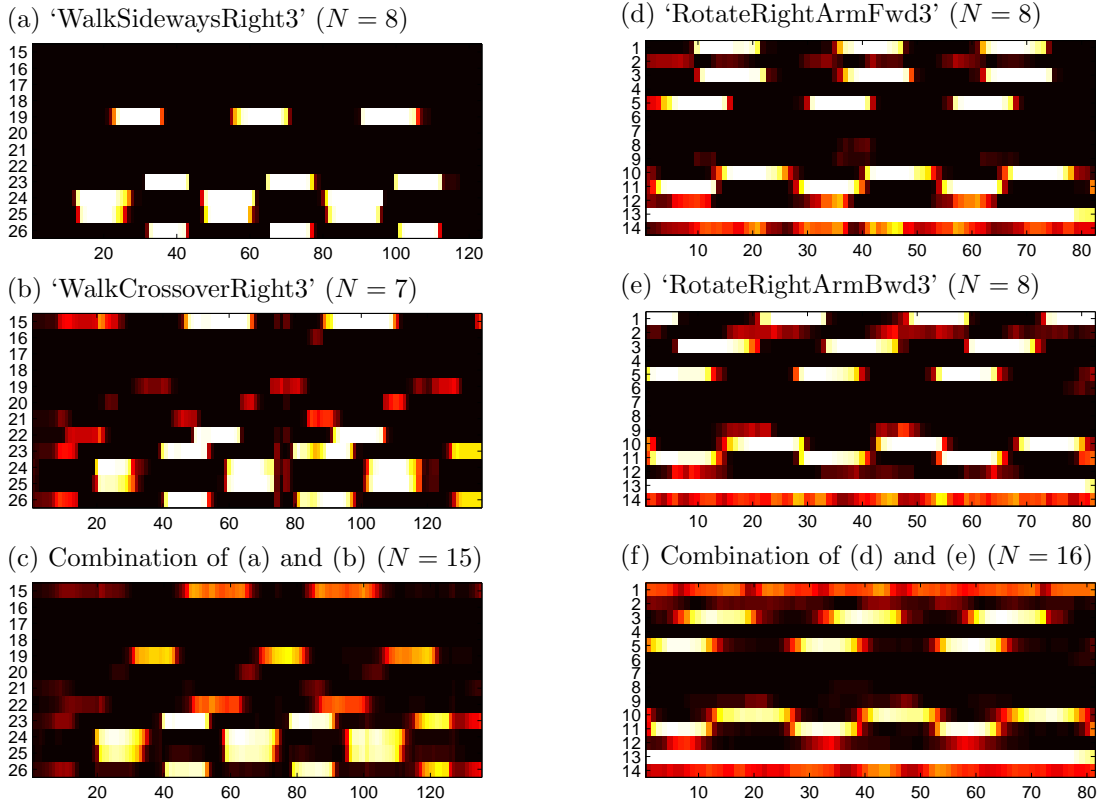


Figure 5.5. **Left column:** class MTs (shown only for the lower feature set) for (a) ‘WalkSidewaysRight3’ with $N = 8$, (b) ‘WalkCrossoverRight3’ with $N = 7$, and (c) the combination of the two classes with $N = 15$ training motions. **Right column:** class MTs (shown only for the upper feature set) for (d) ‘RotateRightArmFwd3’, (e) ‘RotateRightArmBwd3’, and (f) its combination.

the right and the left hand are first moved to the top (F_5 and F_6 assume value one), then remain raised (F_3 and F_4 assume value one), before they drop again at the end. The left and right elbows are not bent throughout the entire movement (F_7 and F_8 only assume the value zero). Furthermore, there are two phases where the hands are apart relative to the shoulders (F_9) and where the hands move together (F_{10}). Also very consistent is the movement of the lower part of the body. For example, the feet first move apart (F_{24}), then are apart (F_{19}), and finally move together (F_{23}). The two phases where both feet are in the air are reflected by the values of the velocity features F_{25} and F_{26} . As for the colored regions, some of the actors performing the motion class ‘JumpingJack’ moved their hands slightly to the front before their hands touched above their head, resulting in some colored values in the rows corresponding to F_1 and F_2 .

The motion classes ‘RotateRArmFwd3’ and ‘RotateRArmBwd3’ stand for three repetitions of forward and backward rotation of the right arm, respectively. They are closely related, even though we do not consider them as logically similar. The respective class MTs are shown for the upper feature set in Fig. 5.5 (a) and (b). Even though the two class MTs exhibit a similar zero-one distribution, there is one characteristic difference: in the forward rotation, the right arm moves forwards (F_1) exactly when it is raised above the shoulder (F_3 is one). By contrast, in the backward rotation, the right arm moves forwards (F_1) exactly when it is

below the shoulder (F_3 is zero). Using the training motions of both classes, it is possible to learn a single, combined MT, see Fig. 5.5 (c). Indeed, the resulting MT very well reflects the common characteristics as well as the disagreements of the two involved classes.

5.2 MT-based Matching for Retrieval and Classification

Given a class \mathcal{C} of logically related motions, we have derived a class MT $X_{\mathcal{C}}$ that captures the consistent as well as the inconsistent aspects of all motions in \mathcal{C} . Our application of MTs to automatic classification and retrieval are based on the following interpretation: the consistent aspects represent the class characteristics that are shared by all motions, whereas the inconsistent aspects represent the class variations that are due to different realizations. Then, the key idea in designing a distance measure for comparing a class MT with unknown motion data is to mask out the inconsistent aspects such that related motions can be identified even in the presence of large variations. In Sect. 5.2.1, we define such a distance measure, which is based on DTW. Our experiments on MT-based classification, annotation, and retrieval are then described in Sect. 5.2.2–5.2.5.

5.2.1 MT-based Matching

In order to compare a class MT with the feature matrix resulting from an unknown mocap data stream, we use a subsequence variant of DTW. The crucial point of our matching strategy is the local cost measure, which disregards the inconsistencies encoded in the class MT. To this end, we introduce a *quantized MT*, which has an entry 0.5 at all positions where the class MT indicates inconsistencies between different executions of a training motion within the same class. More precisely, let δ , $0 \leq \delta < 0.5$, be a suitable threshold. Then for an MT $X \in [0, 1]^{f \times K}$, we define the quantized MT by replacing each entry of X that is below δ by zero, each entry that is above $1 - \delta$ by one, and all remaining entries by 0.5. In our experiments, we used the threshold $\delta = 0.1$. See Fig. 5.3 for an example of a quantized MT.

Now, let D be a mocap data stream. The goal is to identify subsequences of D that are similar to a given motion class \mathcal{C} . Let X be a quantized class MT of length K and Y the feature matrix of D of length L . We define for $k \in [1 : K]$ and $\ell \in [1 : L]$ a local cost measure $c^{\mathcal{Q}}(k, \ell)$ between the vectors $X(k)$ and $Y(\ell)$. Let $I(k) := \{i \in [1 : f] \mid X(k)_i \neq 0.5\}$, where $X(k)_i$ denotes a matrix entry of X for $k \in [1 : K]$, $i \in [1 : f]$. Then, if $|I(k)| > 0$, we set

$$c^{\mathcal{Q}}(k, \ell) = \frac{1}{|I(k)|} \sum_{i \in I(k)} |X(k)_i - Y(\ell)_i|, \quad (5.1)$$

otherwise we set $c^{\mathcal{Q}}(k, \ell) = 0$. In other words, $c^{\mathcal{Q}}(k, \ell)$ only accounts for the consistent entries of X with $X(k)_i \in \{0, 1\}$ and leaves the other entries unconsidered. Furthermore, to avoid degenerations in the DTW alignment, we use the modified step size condition $p_{m+1} - p_m \in \{(2, 1), (1, 2), (1, 1)\}$, cf. condition (iii) of Sect. 2.7. This forces the slope of the warping path to assume values between $\frac{1}{2}$ and 2. Then, the distance function $\Delta_{\mathcal{C}} : [1 : L] \rightarrow \mathbb{R} \cup \{\infty\}$ is defined by¹

$$\Delta_{\mathcal{C}}(\ell) := \frac{1}{K} \min_{a \in [1 : \ell]} \left(\text{DTW}(X, Y(a : \ell)) \right), \quad (5.2)$$

¹Due to the modified step size condition, some of the DTW distances in (5.2) may not exist, which are then set to ∞ .

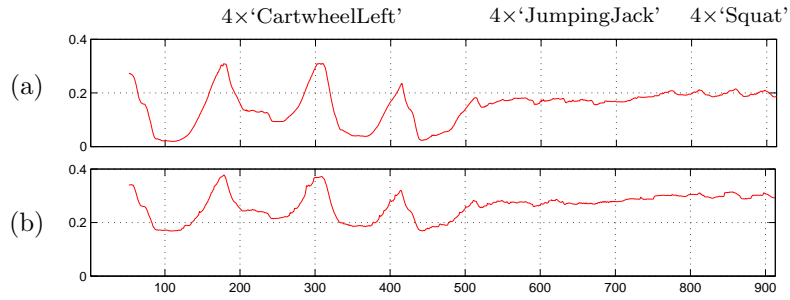


Figure 5.6. (a) Distance function Δ_C based on c^Q of (5.1) for the quantized class MT ‘CartwheelLeft’ and a motion sequence D consisting of four cartwheels (reflected by the four local minima close to zero), four jumping jacks, and four squats. The sampling rate is 30 Hz. (b) Corresponding distance function based on the Manhattan distance without MT quantization, leading to a much poorer result.



Figure 5.7. Resulting distance functions for a 35-second gymnastics sequence (30 Hz) consisting of four jumping jacks, four repetitions of a skiing coordination exercise, two repetitions of an alternating elbow-to-knee motion, and four squats with respect to the quantized class MTs for (a) ‘JumpingJack’, (b) ‘ElbowToKnee’, and (c) ‘Squat’.

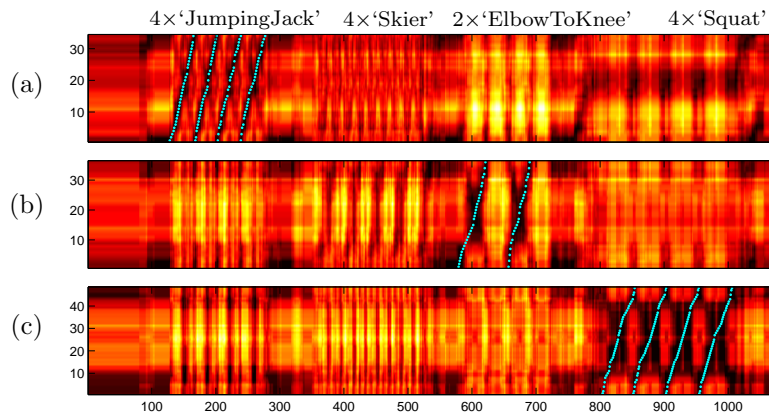


Figure 5.8. Cost matrices corresponding to the distance functions of Fig. 5.7. The warping paths corresponding to the local minima in Fig. 5.7 are indicated as blue lines.

where $Y(a : \ell)$ denotes the submatrix of Y consisting of columns a through $\ell \in [1 : L]$ and the DTW distance $\text{DTW}(X, Y(a : \ell))$ is defined in analogy to Eqn. 2.15. Note that the function $\Delta_{\mathcal{C}}$ can be computed by means of subsequence DTW, see [RJ93]. Furthermore, one can derive from the resulting DTW matrix for each $\ell \in [1 : L]$ the index $a_{\ell} \in [1 : \ell]$ that minimizes (5.2). The interpretation of $\Delta_{\mathcal{C}}$ is as follows: a small value $\Delta_{\mathcal{C}}(\ell)$ for some $\ell \in [1 : L]$ indicates that the subsequence of D starting at frame a_{ℓ} and ending at frame ℓ is similar to the motions of the class \mathcal{C} . Note that using the local cost function c^{Q} of (5.1) based on the quantized MT (instead of simply using the Manhattan distance) is of crucial importance, as illustrated by Fig. 5.6. Here, a gymnastics motion comprising several repetitions of the motion classes ‘CartwheelLeft’, ‘JumpingJack’, and ‘Squat’ is matched against the quantized class MT for ‘CartwheelLeft’ in Fig. 5.6 (a) and against the non-quantized class MT in Fig. 5.6 (b). Observe that the local minima in (a) are much closer to zero than in (b), corresponding to a better separation between the motion classes. This improvement is due to the variational aspects of the cartwheel template being masked out in the quantization step. Further examples are discussed in Sect. 5.2.2.

5.2.2 MT-based Classification

In the classification scenario, we are given an unknown motion data stream D for which the presence of certain motion classes $\mathcal{C}_1, \dots, \mathcal{C}_P$ at certain times is to be detected. These motion classes are identified with their respective quantized class MTs X_1, \dots, X_P , which are assumed to have been precomputed from suitable training data. Now, the idea is to match the input motion D with each of the X_p , $p = 1, \dots, P$, yielding the distance functions $\Delta_p := \Delta_{\mathcal{C}_p}$. Then, every local minimum of Δ_p close to zero indicates a motion subsequence of D that is similar to the motions in \mathcal{C}_p . The subsequences of D corresponding to these minima can then be labeled as belonging to the motion class \mathcal{C}_p . As a result of the entire procedure, certain frame ranges are labeled by certain motion classes, where some frame ranges may have received more than one label and some frame ranges may have not been labeled at all. As an example, we consider the distance functions for a 35-second gymnastics motion sequence with respect to the motion classes $\mathcal{C}_1 = \text{‘JumpingJack’}$, $\mathcal{C}_2 = \text{‘ElbowToKnee’}$, and $\mathcal{C}_3 = \text{‘Squat’}$, see Fig. 5.7. For \mathcal{C}_1 , there are four local minima between frames 100 and 300, which exactly correspond to the four jumping jacks contained in D , see Fig. 5.7 (a). Note that the remaining portion of D is clearly separated by Δ_1 , yielding a value far above 0.1. Analogously, the two minima in Fig. 5.7 (b) and the four minima in Fig. 5.7 (c) correspond to the two repetitions of the elbow-to-knee exercise and the four squats, respectively. The corresponding cost matrices and warping paths are shown in Fig. 5.8. Besides discussing further experiments, we also describe how to choose suitable quality thresholds for Δ_p in the next section.

5.2.3 MT-based Retrieval

The goal of content-based motion retrieval is to automatically extract all logically similar motions of some specified type scattered in a motion capture database \mathcal{D} . By concatenating all documents of \mathcal{D} , we may assume that the database is represented by one single motion data stream D . To retrieve all motions represented by a class \mathcal{C} , we compute the distance function $\Delta_{\mathcal{C}}$ with respect to the precomputed class MT. Then, each local minimum of $\Delta_{\mathcal{C}}$ below some quality threshold $\tau > 0$ indicates a hit. To determine a suitable threshold τ and to measure the retrieval quality, we conducted extensive experiments based on several

databases. We start with the evaluation database \mathcal{D}^{MCE} , which consists of 718 motion clips corresponding to 24 minutes of motion data, see Sect. 5.1.2. Recall that \mathcal{D}^{MCE} is disjoint to the training database \mathcal{D}^{MCT} , from which the class MTs were derived. Fixing a quality threshold τ , we computed a set H_τ of hits for each of the 64 class MTs in a fully automated batch mode. Based on a manually generated annotation of \mathcal{D}^{MCE} used as the ground truth, we then determined the subset $H_\tau^+ \subseteq H_\tau$ of relevant hits corresponding to motion clips of the respective class.

Tables 5.2 and 5.3 show the retrieval results for six different choices of τ . For example, for the motion class ‘ClapHandsAboveHead’ and the quality threshold $\tau = 0.02$, all of the 7 resulting hits are relevant—only one clapping motion is missing. Increasing the quality threshold to $\tau = 0.04$, one obtains 16 hits containing all of the 8 relevant hits. However, one also obtains 8 false positives, mainly coming from the jumping jack class, which contains a similar arm movement. The precision and recall values are very good for whole-body motions such as ‘JumpingJack’, ‘Cartwheel’, or ‘LieDownFloor’—even in the presence of large variations within a class. Short motions with few characteristic aspects such as the class ‘GrabHighRHand’ are more problematic. For $\tau = 0.04$, one obtains 49 hits containing 12 of the 14 relevant movements. Confusion arises mainly with similar classes such as ‘DepositHighRHand’ or ‘GrabMiddleRHand’ and with subsequences of more complex motions containing a grabbing-like component such as the beginning of a cartwheel. Even from a semantical point of view, it is hard to distinguish such motions. Similar confusion arises with increasing values of τ for the kicking, walking/jogging, rotation, or sitting classes. However, most of the relevant hits could be found among the top ranked hits in all cases. For the classes ‘RotateRArmFwd1’ and ‘RotateRArmBwd1’, see Fig. 5.5 (a) and (b), all relevant movements could be correctly identified. Using a combined MT, as indicated by Fig. 5.5 (c), the two classes could not be distinguished any longer—the characteristics that had separated the two classes were now regarded as mere variations and therefore masked out in the retrieval process.

The above experiments imply that the quality threshold $\tau = 0.06$ constitutes a good trade-off between precision and recall. Since the distance function Δ_C yields a ranking of the retrieved hits in a natural way, our strategy is to allow for some false positives rather than having too many false negatives. Furthermore, note that the running time and memory requirements of MT-based retrieval depends linearly on the size of the database, where the bottleneck is the computation of the distance function Δ_C . For example, in case of the 24-minute database \mathcal{D}^{MCE} , our Matlab implementation required 4–28 seconds to process one query—depending on the respective MT length and the number of hits, see Tables 5.2 and 5.3.

5.2.4 Keyframes

To speed up the retrieval on large databases and to significantly reduce memory requirements, we introduce an efficient preselection step to cut down the set of candidate motions prior to computing Δ_C . To this end, we label a small number of characteristic columns of each class MT as *keyframes*. The underlying assumption is that most instances of a motion belonging to the respective motion class will exhibit the feature vectors pertaining to the keyframes in exactly the sequence denoted by the keyframe positions. In our experiments, the keyframes were picked automatically using a simple heuristic: we basically chose two to five columns from the quantized MT that had many “white” entries (i. e., entries close to one, indicating some consistent action) and few “gray” entries (i. e., entries indicating inconsistencies).

Motion Class \mathcal{C}	γ	$ H_\tau $ / $ H_\tau^+ $						K	$t(\Delta_{\mathcal{C}})$
CartwheelLeft	10	1	4	6	8	9	10	106	12.97
ClapHandsAboveHead	8	5	7	16	39	61	81	25	4.14
DepositFloorR	16	4	7	15	31	48	70	73	9.59
DepositHighR	14	11	21	47	60	103	197	67	11.39
DepositLowR	14	8	10	12	13	13	13	68	10.56
DepositMiddleR	14	21	52	122	180	236	295	74	14.61
ElbowToKnee	13	8	11	12	13	13	22	36	4.19
GrabFloorRHand	8	6	8	11	20	41	75	61	8.36
GrabHighRHand	14	15	22	49	58	115	201	68	11.39
GrabLowR	14	14	20	33	55	88	150	73	10.95
GrabMiddleR	14	20	71	150	227	323	408	59	14.77
HitRHandHead	6	29	70	152	257	343	432	55	16.83
HopBothLegs	18	13	19	22	32	126	334	24	6.56
HopLLeg	20	15	19	23	51	96	174	19	3.83
HopRLeg	21	17	19	22	35	66	107	18	3.08
JogLeftCircleRFootStart4	8	7	20	39	43	63	84	60	8.00
JogRightCircleRFootStart4	8	6	13	37	41	53	74	59	7.73
JumpDown	7	3	3	6	6	10	46	66	8.28
JumpingJack	26	25	26	26	26	33	40	34	4.11
KickFrontLFoot	14	4	8	23	126	328	465	54	13.64
KickFrontRFoot	15	3	6	26	90	239	385	54	13.70
KickSideLFoot	13	2	6	16	26	87	299	55	11.61
KickSideRFoot	15	6	13	27	48	163	359	51	12.88
LieDownFloor	10	4	6	8	8	9	11	172	28.05
PunchFrontLHand	15	1	5	22	42	173	414	48	12.77
PunchFrontRHand	15	4	6	27	92	289	442	54	13.70
PunchSideLHand	15	7	13	24	48	229	500	41	13.05
PunchSideRHand	14	2	9	40	104	355	500	41	12.69
RotateBothArmsBwd1	8	7	7	8	8	11	30	29	3.47
RotateBothArmsFwd1	8	8	8	8	8	15	47	30	3.69
RotateLArmBwd1	8	5	6	10	14	51	157	29	5.34
RotateLArmFwd1	8	5	6	9	21	55	176	28	5.67
RotateRArmBwd1	8	6	6	7	34	70	151	27	5.16
RotateRArmFwd1	8	6	6	7	39	77	186	28	6.13

Table 5.2. First part of the retrieval results for the evaluation database \mathcal{D}^{MCE} for various class MTs. Note that \mathcal{D}^{MCE} is disjoint to the training database \mathcal{D}^{MCT} , from which the class MTs were derived. γ denotes the number of relevant motions contained in \mathcal{D}^{MCE} . $|H_\tau|$ (first rows) denotes the number of hits and $|H_\tau^+|$ (second rows) the number of relevant hits with respect to the quality thresholds $\tau = 0.01, 0.02, 0.04, 0.06, 0.08, \text{ and } 0.1$. Finally, K denotes the length of the class MT and $t(\Delta_{\mathcal{C}})$ the running time in seconds required to compute the respective distance function $\Delta_{\mathcal{C}}$.

Motion Class \mathcal{C}	γ	H_τ		H_τ^+		K	$t(\Delta_{\mathcal{C}})$		
RotateRArm(Bwd&Fwd)1	16	12 12	12 12	39 13	101 15	235 16	453 16	26	9.95
Shuffle2StepsLStart	6	11 5	20 6	57 6	122 6	255 6	383 6	48	13.22
Shuffle2StepsRStart	6	15 5	25 5	67 6	152 6	228 6	295 6	62	13.66
SitDownChair	10	4 4	9 8	17 10	29 10	53 10	70 10	83	12.92
SitDownFloor	10	9 4	15 6	25 9	34 10	48 10	61 10	106	15.78
SitDownKneelTieShoes	8	5 5	7 7	8 8	9 8	13 8	20 8	166	22.16
SitDownTable	10	19 9	31 10	80 10	125 10	184 10	260 10	69	15.22
Skier	15	12 12	13 13	15 15	16 15	25 15	56 15	36	4.80
Squat	26	23 23	24 24	26 26	26 26	26 26	27 26	48	5.69
StaircaseUp3	14	9 9	11 11	18 13	32 13	59 14	143 14	77	11.64
StandUpKneelToStand	8	4 4	6 5	15 7	26 7	53 7	89 7	49	6.86
StandUpLieFloor	10	3 3	7 7	10 8	14 8	18 10	23 10	129	16.47
StandUpSitChair	10	7 7	15 10	18 10	32 10	50 10	74 10	70	9.33
StandUpSitFloor	10	9 6	10 6	18 8	25 10	39 10	58 10	94	12.44
StandUpSitTable	10	29 10	53 10	104 10	191 10	266 10	345 10	59	15.50
ThrowBasketball	7	3 3	5 5	5 5	16 7	38 7	68 7	94	12.33
ThrowFarR	7	3 3	8 7	11 7	20 7	62 7	92 7	128	17.61
ThrowSittingHighR	7	3 3	7 6	11 7	14 7	19 7	29 7	72	9.19
ThrowSittingLowR	7	3 3	7 7	11 7	14 7	31 7	53 7	66	9.27
ThrowStandingHighR	7	4 4	5 5	22 6	48 7	129 7	210 7	78	13.39
ThrowStandingLowR	7	5 5	7 6	18 7	128 7	247 7	340 7	81	15.59
TurnLeft	15	37 12	72 15	195 15	310 15	397 15	489 15	49	16.50
TurnRight	15	39 11	81 13	174 14	261 15	344 15	427 15	54	16.38
WalkFwdRFootStart4	8	17 7	21 7	25 8	44 8	69 8	131 8	82	11.42
WalkBwdRFootStart4	7	6 6	7 7	7 7	24 7	72 7	101 7	97	13.41
WalkCrossoverRight3	6	6 6	7 6	14 6	24 6	32 6	56 6	136	17.87
WalkSidewaysRight3	8	7 7	8 8	11 8	14 8	28 8	49 8	123	16.08
WalkLeftCircle4	9	13 7	21 8	24 9	41 9	73 9	118 9	83	11.61
WalkRightCircle4	7	14 5	18 6	22 7	41 7	67 7	129 7	84	11.78
Walk(Crossover & Sideways)Right3	14	13 13	14 14	18 14	29 14	49 14	69 14	135	17.95

Table 5.3. Second part of the retrieval results for the evaluation database \mathcal{D}^{MCE} for various class MTs. See Tab. 5.2 for an explanation.

Then, as a preprocessing step, we extract those segments from the unknown motion database D that exhibit feature vectors matching the specified keyframes in the correct order within suitable time bounds. More precisely, let $v_1, \dots, v_J \in \{0, 0.5, 1\}^f$ be the sequence of selected keyframe columns from the quantized MT, and let $Y \in \{0, 1\}^{f \times K}$ be the feature matrix for D . We are looking for minimal integer intervals $[a : b] \subseteq [1 : K]$ such that there exists an ascending sequence of frame numbers $a = k_1 < k_2 < \dots < k_J = b$ satisfying

$$\forall j \in [1 : J] : Y(k_j) \in v_j \quad \wedge \quad k_J - k_1 + 1 \leq T, \quad (5.3)$$

where for $y \in \{0, 1\}^f$ and $v \in \{0, 0.5, 1\}^f$ we define

$$y \subseteq v \Leftrightarrow y^{I(v)} = v^{I(v)}, \quad (5.4)$$

with $I(v) := \{i \in [1 : f] \mid v^i \neq 0.5\}$, similar to (5.1). In other words, the comparison of y and v ignores the “gray” entries ($v^i = 0.5$). The threshold $T > 0$ denotes the maximum admissible temporal separation between the occurrence of the first and the last keyframe.

This preselection can be done efficiently using inverted lists as described in Sect. 4.2.1. Similar to (4.10), we construct for each v_j , $j \in [1 : J]$, a corresponding fuzzy set

$$V_j := \left\{ v \in \{0, 1\}^f \mid v \subseteq v_j \right\}. \quad (5.5)$$

As in the case of adaptive fuzzy queries (Sect. 4.1.3), we assume that (V_1, \dots, V_J) is an admissible fuzzy query, where neighboring fuzzy sets are disjoint. Similar to (4.8), let $L_j := L(V_j)$ be the sorted union of inverted lists corresponding to V_j . Since we are interested in frame numbers rather than segment numbers (as stored in the inverted lists), we use the additional information about segment boundaries stored along with the index to convert the inverted lists L_j into suitable lists of frame numbers. For any segment number h pertaining to the F -segmentation of D , let $s(h)$ and $e(h)$ denote the corresponding start and end frames, respectively. Writing the j^{th} inverted list as $L_j = (h_j^1, \dots, h_j^{\ell_j})$, we first define the first list of representative frame numbers

$$\Lambda_1 := (e(h_1^1), \dots, e(h_1^{\ell_1})), \quad (5.6)$$

containing the *end frames* of the segments corresponding to L_1 . The remaining lists Λ_j for $2 \leq j \leq J$ are defined as

$$\Lambda_j := (s(h_j^1), \dots, s(h_j^{\ell_j})), \quad (5.7)$$

containing the *start frames* of the segments corresponding to L_j . We make this distinction for the following reason: recall that we are interested in finding minimal integer intervals $[a : b] \subseteq [1 : K]$ and corresponding sequences $a = k_1 < k_2 < \dots < k_J = b$ satisfying the property (5.3). For any such minimal sequence, k_1 will be the end frame of a segment matching v_1 , and k_J will be the start frame of a segment matching v_J . Hence, it is sufficient to restrict our search to such frames.

Next, we perform a linear, simultaneous sweep through the Λ_j using the pointers p_1, \dots, p_J , which are initialized to 1. The first candidate for a start index is $a = k_1 := \Lambda_1(p_1)$, since, by construction, $Y(k_1) \subseteq v_1$. Incrementing the list pointer p_2 , we then search the second inverted list for a candidate position $k_2 := \Lambda_2(p_2)$ with $k_2 > k_1$ satisfying $|k_2 - k_1| \leq T$. If such a k_2 exists, we continue with the third list in the same fashion, and so on. In case a sequence of indices $k_1 < k_2 < \dots < k_J$ can be constructed in this way, we report a match $[a : b] = [k_1 : k_J]$, increment p_1 , and then restart the procedure at Λ_1 . If at any point during the procedure a suitable k_j cannot be found, we know that the current k_1 cannot be the starting position a of an interval $[a : b]$ satisfying (5.3). The search is therefore restarted with a new k_1 . If at any point a list pointer passes the end of its list, the procedure is terminated.

The matched intervals may not be disjoint. For example, this can be observed in periodic motions such as walking. To minimize the amount of data that has to be processed by the subsequent DTW matching, we perform another linear postprocessing sweep through the matched intervals, conjoining overlapping intervals. To account for possible global differences

(a) Automatic keyframe selection

Motion Class \mathcal{C}	#(kf)	sel (m)	sel (%)	$t(\text{kf})$	γ	$ H_{0.06} $	$ H_{0.06}^+ $	$t(\Delta_{\mathcal{C}})$
CartwheelLeft	2	2.8	1.3%	0.02	24	21	20	0.83
ElbowToKnee	2	0.8	0.4%	0.03	29	16	16	0.13
GrabHighRHand	2	8.9	4.2%	0.14	30	128	30	2.77
HopRightLeg	2	1.7	0.8%	0.25	75	81	64	0.27
JumpingJack	2	1.5	0.7%	0.09	52	50	50	0.19
KickFrontRFoot	2	3.2	1.5%	0.03	38	73	36	0.58
KickSideRFoot	2	2.2	1.0%	0.11	38	46	38	0.34
LieDownFloor	2	15.3	7.2%	0.06	20	24	16	4.42
RotateRArmFwd1	2	0.5	0.2%	0.48	66	6	5	0.17
SitDownChair	2	16.2	7.6%	0.11	20	27	4	3.00
Squat	2	2.2	1.1%	0.08	56	55	55	0.33
WalkCrossoverRight3	2	12.9	6.1%	0.03	16	137	16	4.33

(b) Manual keyframe selection

Motion Class \mathcal{C}	#(kf)	sel (m)	sel (%)	$t(\text{kf})$	γ	$ H_{0.06} $	$ H_{0.06}^+ $	$t(\Delta_{\mathcal{C}})$
GrabHighRHand	3	3.2	1.5%	0.16	30	59	30	1.08
LieDownFloor	3	6.5	3.1%	2.75	20	19	19	2.86
RotateRArmFwd1	3	1.0	0.5%	0.33	66	32	32	0.63
SitDownChair	3	3.8	1.8%	0.17	20	34	16	1.28
WalkCrossoverRight3	5	4.9	2.3%	0.14	16	66	16	2.06

Table 5.4. Upper: Retrieval results for the database \mathcal{D}_{210} and $\tau = 0.06$ based on automatic keyframe selection. The second to fourth columns indicate the number of keyframes, the size of the preselected data set in minutes and percent as well as the running time for the preprocessing step. γ is the number of relevant motions in \mathcal{D}_{210} . $|H_{0.06}|$ and $|H_{0.06}^+|$ denote the number of hits and the number of relevant hits, respectively. $t(\Delta_{\mathcal{C}})$ indicates the running time in seconds required to compute $\Delta_{\mathcal{C}}$ on the preselected motions. **Lower:** Retrieval results for manually selected keyframes.

in motion speed between the MT and the unknown motion, we further extend the resulting intervals by a certain number of frames ℓ and r to the left and to the right, respectively. Assuming that the keyframes v_1, \dots, v_J correspond to columns $X(i_1), \dots, X(i_J)$ from an MT $X \in [0, 1]^{f \times K}$, we choose $\ell := 3i_1$ and $r := 3(K - i_J + 1)$. Furthermore, we choose the interval length threshold $T := 2(i_J - i_1 + 1)$, accounting for motions that are slower than the template by up to a factor of 2.

Once a set of suitable motion segments has been precomputed in this way, we compute the distance function $\Delta_{\mathcal{C}}$ only on those motion segments. This strategy has been applied to our (unlabeled) 210-minute database \mathcal{D}_{210} , which was introduced in Sect. 5.1.2. Some retrieval results as well as running times are summarized in Table 5.4 (a). To assess retrieval quality, we manually inspected the set $H_{0.06}$ of hits as well as the database \mathcal{D}_{210} for each class to determine the set $H_{0.06}^+$ of relevant hits. For example, the database \mathcal{D}_{210} contains 24 left cartwheels. Using two automatically determined keyframes, it took 20 milliseconds to reduce the data from 210 to 2.8 minutes—1.3% of the original data. Then, MT retrieval was performed on the preselected 2.8 minutes of motion, which resulted in 21 hits and took 0.83 seconds. These hits contained 20 of the 24 cartwheels.

Even though keyframes are a powerful tool to significantly cut down the search space, there is also an attached risk: a single inappropriate keyframe may suffice to produce a large number of false negatives. For example, this happened for the classes listed in Table 5.4 (b).

For these classes, using more appropriate, manually selected keyframes led to a significant improvement. A further benefit of the keyframe approach is that the large number of false positives, as typical for short and unspecific motions, can be easily cut down by adding a single keyframe. See, for example, the motion class ‘GrabHighRHand’ in Table 5.4 (a).

As a further test, we used the 180-minute database $\mathcal{D}_{180}^{\text{CMU}}$ containing motion capture material from the CMU database [CMU03]. Similar results and problems can be reported as for \mathcal{D}_{210} . Interestingly, our class MT X for ‘CartwheelLeft’ yielded no hits at all—as it turned out, all cartwheels in $\mathcal{D}_{180}^{\text{CMU}}$ are right cartwheels. We modified X by simply interchanging the rows corresponding to feature pairs pertaining to the right/left part of the body, see Table 3.2. Using the resulting *mirrored* MT, four out of the known five cartwheels in $\mathcal{D}_{180}^{\text{CMU}}$ appeared as the only hits. As a second interesting result, no relevant hits were reported as top hits for our ‘Squat’ MT. A manual inspection showed that for all squatting motions of $\mathcal{D}_{180}^{\text{CMU}}$ the arms are kept close to the body, whereas in the training motions the arms are consistently stretched out in front of the body. Manually modifying the class MT by masking out the features for the arm motions, we were able to retrieve most squatting motions from $\mathcal{D}_{180}^{\text{CMU}}$. In other words, the class MTs are invariant only under those variations that are reflected by the training motions. Due to their semantic meaning, class MTs can easily be modified in an intuitive way without any additional training data. Even designing a class MT from scratch (without resorting to any training motions) proved to be feasible. For example, to identify ‘sweeping with a hand brush’ in $\mathcal{D}_{180}^{\text{CMU}}$, we defined an MT of length 50, setting all matrix entries to 0.5 except for the rows corresponding to F_{13} (right hand fast), F_{32} (spine horizontal), and F_{33} (right hand lowered), which were set to one. Eight out of ten hits in $\mathcal{D}_{180}^{\text{CMU}}$ were relevant.

5.2.5 Comparison to Other Retrieval Methods

We compared our MT-based retrieval system to several baseline methods using subsequence DTW on raw motion capture data with suitable local distance measures. It turned out that such baseline methods show little or no generalization capability. The database (3.8 minutes, or 6,750 frames sampled at 30 Hz) consisted of 100 motion clips: ten different realizations for each of ten different motion classes. For each of the ten motion classes, we performed motion retrieval in four different ways:

- (MT) retrieval using a quantized class MT,
- (RF) DTW using the relational feature matrix of a single example motion and Manhattan distance,
- (Q) DTW using unit quaternions and the cost measure c^{quat} , and
- (3D) DTW using 3D joint coordinates (normalized w. r. t. root rotation and size) and Euclidean distance.

For each strategy, we computed a Δ curve as in Fig. 5.6 and derived the top 5, top 10, and top 20 hits. Table 5.5 shows the resulting recall values (note that there are exactly 10 correct hits for each class) for five representative queries. As a further important quality measure of a strategy, we computed the *separation quotient*, denoted by s , which is defined as the median of Δ divided by the median of the cost of the correct hits among the top 10 hits. The larger the

Motion Class	$r_{5/10/20}^{\text{MT}}$	s^{MT}	$r_{5/10/20}^{\text{RF}}$	s^{RF}	$r_{5/10/20}^{\text{Q}}$	s^{Q}	$r_{5/10/20}^{\text{3D}}$	s^{3D}
CartwheelLeft	5/10/10	12.83	5/10/10	1.62	4/6/7	1.63	1/1/2	2.38
Squat	5/10/10	259.5	5/10/10	16.1	5/7/9	2.79	4/6/7	2.52
LieDownFloor	5/9/10	11.65	5/9/10	2.10	4/7/9	1.69	2/3/7	1.29
SitDownFloor	4/6/10	19.33	3/4/8	1.60	2/5/7	2.13	3/5/8	1.56
GrabHighRHand	5/7/9	33.93	5/8/8	9.72	3/5/8	3.39	1/3/4	2.22

Table 5.5. Recall values (r) in the top 5/10/20 ranked hits and separation quotients (s) for different DTW-based retrieval methods: motion templates (MT), relational feature matrices (RF), quaternions (Q), and relative 3D coordinates (3D).

value of s , the better the correct hits are separated from the false positives, enabling the usage of simple thresholding strategies on Δ for the retrieval. Only for our MT-based strategy, the separation is good enough. These observations indicate that MT-based retrieval significantly outperforms the other methods. More results can be found at our web site [MR06a].

We also compared MT-based retrieval to adaptive fuzzy querying as introduced in Chap. 4. Recall that the performance of adaptive fuzzy querying heavily depends on the query formulation, which involves manual specification of a query-dependent feature selection. For each query, we carefully selected a suitable subset of features. The resulting precision/recall values on \mathcal{D}^{MC} are very good and reflect what seems to be achievable by adaptive fuzzy querying, see Table 5.6. For MT-based retrieval, we quote precision/recall values for two quality thresholds, $\tau = 0.02$ and $\tau = 0.06$. Our experiments show that the retrieval quality of our fully automatic MT-based approach is in most cases as good and in many cases even better than that obtained by adaptive fuzzy querying, even after hand-tweaking the feature selection. Hence, our MT-based approach enables us to replace manual, global feature selection by fully automatic, local feature selection without loss of retrieval quality.

5.3 Related Work

Our MT-based approach to motion classification differs from other techniques in that we work with an explicit motion representation learned exclusively from *positive* training examples describing the respective motion class. This is in contrast to typical supervised learning techniques such as support vector machines (SVM) [DHS00], where classifiers are trained from examples that are potentially labeled by many different classes. For SVM-based classifiers, changing or adding only one training example may have an impact on the classification performance for all classes known to the classifier—that is, the classifier implicitly uses the information that there is a limited, known number of classes. As for MTs, only the single MT corresponding to the respective class changes if a training motion is changed or added. Each MT can then serve as a detector for the presence of a single motion class. Consequently, a given set of MTs does not define a partition on the set of all possible motions: a motion can be matched by many MTs. Having this in mind, we review some of the recent work on motion classification, recognition, and annotation.

Arikan and Forsyth [AFO03]. Arikan and Forsyth propose a technique to synthesize realistic motion sequences from user-specified, textual action commands where automatically selected bits of a motion database are blended together. In a preprocessing step, each frame of



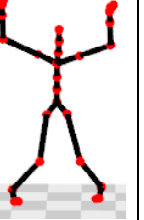





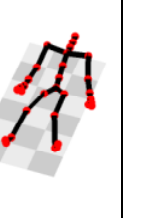



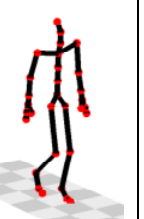
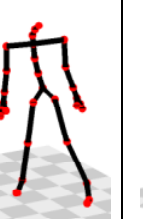
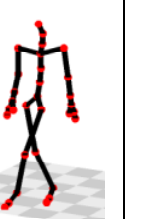
	elbow-to-knee	cartwheel	jumping jack	hop both legs	hop right leg
					
Adaptive fuzzy					
recall	24/27 = 0.89	21/21 = 1.00	51/52 = 0.98	21/36 = 0.58	33/42 = 0.79
precision	24/26 = 0.92	21/21 = 1.00	51/55 = 0.93	21/34 = 0.62	33/182 = 0.18
ranking top 5 10 20	5 10 20	5 10 20	5 10 20	4 8 16	5 10 20
MT-based retrieval					
recall ($\tau = 0.02$)	23/27 = 0.85	12/21 = 0.57	51/52 = 0.98	34/36 = 0.94	40/42 = 0.95
precision ($\tau = 0.02$)	23/23 = 1.00	12/12 = 1.00	51/51 = 1.00	34/38 = 0.89	40/40 = 1.00
recall ($\tau = 0.06$)	27/27 = 1.00	19/21 = 0.90	52/52 = 1.00	36/36 = 1.00	42/42 = 1.00
precision ($\tau = 0.06$)	27/27 = 1.00	19/19 = 1.00	52/52 = 1.00	36/69 = 0.52	42/73 = 0.57
	hit on head	kick right	sit down	lie down	rotate both arms
					
Adaptive fuzzy					
recall	12/13 = 0.92	25/30 = 0.83	16/20 = 0.80	19/20 = 0.95	16/16 = 1.00
precision	12/14 = 0.86	25/41 = 0.61	16/40 = 0.40	19/21 = 0.90	16/16 = 1.00
ranking top 5 10 20	5 9 12	5 8 15	5 7 10	5 10 18	5 10 16
MT-based retrieval					
recall ($\tau = 0.02$)	12/13 = 0.92	14/30 = 0.47	14/20 = 0.70	16/20 = 0.80	16/16 = 1.00
precision ($\tau = 0.02$)	12/158 = 0.08	14/18 = 0.77	14/14 = 1.00	16/19 = 0.84	16/16 = 1.00
recall ($\tau = 0.06$)	13/13 = 1.00	26/30 = 0.87	17/20 = 0.85	19/20 = 0.95	16/16 = 1.00
precision ($\tau = 0.06$)	13/500 = 0.03	26/196 = 0.13	17/21 = 0.81	19/64 = 0.30	16/16 = 1.00
	walk up staircase	walk	walk backwards	walk sideways	walk cross over
					
Adaptive fuzzy					
recall	22/28 = 0.79	15/16 = 0.94	8/15 = 0.53	16/16 = 1.00	10/13 = 0.77
precision	22/23 = 0.96	15/33 = 0.45	8/22 = 0.36	16/17 = 0.94	10/13 = 0.77
ranking top 5 10 20	5 10 20	2 6 12	4 8 8	5 10 16	5 10 10
MT-based retrieval					
recall ($\tau = 0.02$)	22/28 = 0.76	15/16 = 0.94	15/15 = 1.00	16/16 = 1.00	13/13 = 1.00
precision ($\tau = 0.02$)	22/22 = 1.00	15/45 = 0.33	15/15 = 1.00	16/16 = 1.00	13/13 = 1.00
recall ($\tau = 0.06$)	27/28 = 0.96	16/16 = 1.00	15/15 = 1.00	16/16 = 1.00	13/13 = 1.00
precision ($\tau = 0.06$)	27/67 = 0.40	16/88 = 0.18	15/57 = 0.26	16/19 = 0.84	13/48 = 0.27

Table 5.6. Comparison of hand-tuned fuzzy queries and MT-based queries on a subset of \mathcal{D}^{MC} .

the motion database has been assigned to one or several predefined motion classes. This assignment is performed by a semi-automatic annotation procedure using support vector machine classifiers based on 3D trajectory data. Ramanan and Forsyth [RF03] also apply this annotation technique for 3D motion data as a preprocessing step for automatic annotation of 2D video recordings of human motion, using hidden Markov models (HMMs) to match the 2D data with the 3D data.

Green and Guan [GG04]. Inspired by speech recognition with HMMs, Green and Guan [GG04] use progressions of kinematic features such as motion vectors of selected joints, so-called *dynemes*, for their video-based motion recognition system. Before a motion class can be recognized, the structure of the HMM for that motion class has to be manually assembled from a set of given building blocks and then has to be trained with suitable training data. See also Sect. 2.3 for a discussion of their technique.

Gesture, dance, and gait recognition. Wilson and Bobick [WB99] apply HMMs for video-based gesture recognition. Their main contribution is a parametrization of the HMM's output probabilities that enables certain aspects of gestures to be controlled by continuous parameters. As an example, they quote the gesture of “showing off the size of a fish that one caught”. Here, the distance of the hands during the performance of the gesture has an important meaning, which can be captured by their model. Opposed to such HMM-based motion representations, where timing information is encoded in the form of transition probabilities, MTs encode absolute and relative lengths of key events explicitly.

The motion recognition system described by Campbell and Bobick [CB95] is based on a joint angle representation and has been applied to recognize the presence of nine different types of ballet moves from unsegmented motion data streams. Similar to MTs, their technique may also detect the presence of several motion classes at the same time.

Bissacco et al. [BCMS01] work with state-space (ARMA) models of 3D joint angles derived from 2D video data to roughly distinguish three different types of gait (“walking”, “running”, “climbing stairs”).

Temporal segmentation. Temporal segmentation of motion data, i. e., the process of breaking up continuous mocap data streams into groups of consecutive, logically related frames, is another important task. Pure segmentation techniques, as opposed to classification/recognition techniques, do not assign class labels or *meaning* to motion subclips. Typical examples are the methods by Fod et al. [FMJ02] and Barbic et al. [BSP⁺04].

Chapter 6

Conclusions and Future Work

Motivated by the problem of identifying logically related motions even in the presence of significant spatio-temporal variations, we have introduced a new class of boolean, relational features—opposed to quantitative, numerical features used in previous approaches. These features exploit a key property of mocap data, namely that each joint in a mocap data stream has an explicit meaning. Our features are designed to inherit these explicit semantics, thereby providing a basis for intuitive query formulation. While suitable combinations of relational features are capable of absorbing spatial variations, we have seen that our concept of temporal segmentation can absorb temporal variations that may distinguish similar motions. Together, the concepts of relational motion features and feature-driven temporal segmentation become a powerful tool for flexible, efficient, and automatic content-based motion retrieval and classification.

Extending the notion of fault-tolerant fuzzy retrieval, we have proposed the concept of adaptive fuzzy search to efficiently query large databases using the query-by-example mode. Adaptive fuzzy search allows for the segmentation of the motion database to be automatically adjusted to the fuzziness of the query *at query time*. The underlying index structure based on inverted lists is very simple and only needs to be precomputed once. Furthermore, the space and time complexity of indexing is linear in the size of the database. This solves the problem of scalability emerging in DTW-based approaches such as the match web, see [KG04]. While our methods are tuned towards generating false positive matches rather than false dismissals, we have seen that simple ranking techniques can separate the correct hits from the false positives. We have also sketched how our methods can be applied to accelerate DTW-based time alignment of motion capture data streams: using simple boolean relations, a rough pre-alignment can be computed that can then be refined by means of DTW.

As a drawback of the index-based retrieval approach, we have seen that motion variations such as “insertions” or “deletions” of certain events cannot be handled, which is the price that one pays for efficiency. As a further disadvantage in view of automation, our retrieval scenario requires the user to select suitable features for each query in order to obtain high-quality retrieval results. This is not feasible in case many different motion clips are to be batch processed, as necessary in morphing and blending applications. We have sketched a combinatorial method of trying out different feature functions, efficiently retrieving the corresponding hits, and ranking the results.

Our final contribution was the introduction of motion templates and related techniques. An MT encodes the characteristic as well as the variable aspects of a motion class in a single,

compact matrix representation. We proposed an automatic procedure to learn a class MT from example motions. We then applied class MTs to motion classification and retrieval. By automatically masking out the variable aspects of a motion class in the retrieval process, related motions can be automatically identified even in the presence of large variations and without any user intervention. Extensive experimental results show that our methods work with high precision and recall for whole-body motions and for longer motions of at least a second. More problematic are very short and unspecific motion fragments. Here, the use of suitably defined keyframes is a promising concept to not only speed up the retrieval process, but also to eliminate false positives.

6.1 Future work

So far, our relational features have been manually designed. Automatic, combinatorial methods using standard data analysis techniques such as PCA did not result in useful features due to the lack of semantics of arbitrary combinations of, for example, plane-defining joints. Here, suitable alternatives facilitating automatic feature design have to be found. Somehow, the statistics of a training motion database have to be incorporated in the feature design process in a meaningful way.

Despite providing flexible fault-tolerance mechanisms, our index-based retrieval algorithms still lead to a rather strict notion of motion similarity due to inherent limitations. In the spirit of keyframes as discussed in Sect. 5.2.4, we plan to further investigate combined techniques involving efficient index-based preselection and DTW-based methods. We have seen that a single misplaced keyframe can lead to a large number of false dismissals. Hence, our heuristic method of keyframe selection needs to be improved, incorporating more factors than just the number of consistent entries contained in a column of an MT. Here, it might be possible to use similar methods as described in [ACCO05], or genetic algorithms [Dem06].

We plan to extend our concepts, which are currently based on skeleton-based mocap data, towards pure 3D trajectory data without bone length constraints (such as the widely used C3D format), so as to make our techniques more readily applicable in computer animation and computer vision. In collaboration with the HdM school of media sciences (Stuttgart), we also plan to investigate how motion templates may be used as a tool for specifying animations—replacing a keyframe-based by a template-based animation concept—similar to approaches such as [RCB98, AFO03]. Further ideas regarding possible applications of relational features and the MT concept involve motion classification scenarios appearing in fields such as sports performance analysis, rehabilitation, or the stabilization and enhancement of markerless, video-based motion capture [CH05].

As a theoretical issue, the convergence of the DTW-based warping and averaging algorithm for MT computation has so far not been proven. Also, it would be interesting to assess the suitability of this algorithm for multiple alignment problems. We are also planning to compare motion templates with HMMs, and to compare the classification performance of MTs to machine learning techniques such as SVMs.

Finally, it would be extremely valuable to the research community if publicly available motion databases such as CMU [CMU03] were annotated by motion class labels at the frame level, thus making experimental results comparable. For example, lacking such a common database, it was not possible for us to objectively compare and combine our retrieval results with those of related techniques such as the match web by Kovar and Gleicher [KG04].

Appendix A

Representing Rotations of \mathbb{R}^3

We have seen in Sect. 1.3 that rotations of three-dimensional Euclidean space play an important role in encoding motions via kinematic chains and forward kinematics (FK). Beyond FK, various other problems involving kinematic chains are of interest, for example inverse kinematics (IK), forward dynamics (FD), and inverse dynamics (ID). Given a fixed skeleton and a set of joint positions and rotations, IK deals with the problem of finding suitable values for the skeleton’s free parameters. In analogy, FD and ID deal with the problem of relating physical parameters such as time-dependent forces and torques to time-dependent joint positions and rotations for a fixed skeleton. Such problems are typically solved by means of optimization and interpolation methods involving rotations, thus requiring different kinds of rotational representations. Apart from FK, IK, FD, and ID, a large number of motion editing, warping, and blending techniques work with different rotational representations, for example [BW95, Pop99, PB02]. This also holds true for numerical comparison of mocap data, see [HPP05, SKK04, WCYL03]. Based in parts on [Fau01, För98, Gra98, Koe92, Sch97, Sho85], this section presents some of the associated rotational representations and provides a basis for Appendix D, where we give an overview of common mocap file formats.

Fundamental Properties of Rotations. A 3D rotation is a linear, bijective mapping of \mathbb{R}^3 onto itself that preserves angles, lengths, and “handedness”, thus mapping right-handed orthonormal bases into right-handed orthonormal bases. In referring to the sign of the determinant, we deliberately use the term “handedness” instead of “orientation” since the term orientation is used in the graphics and vision communities to denote the rotational positioning of an object in 3D, see [Fau01]. In this sense, rotations describe orientations in 3D. From an abstract point of view, the set of all 3D rotations forms a subgroup of the automorphism group $\text{Aut}(\mathbb{R}^3)$, the so-called *special orthogonal group* of \mathbb{R}^3 ,

$$\text{SO}(\mathbb{R}^3) = \{F \in \text{Aut}(\mathbb{R}^3) \mid F^* = F^{-1}, \det F = 1\}, \quad (\text{A.1})$$

where F^* is the adjoint of F . $\text{SO}(\mathbb{R}^3)$ is a three-dimensional Lie group. Unlike $\text{SO}(\mathbb{R}^2)$, the group $\text{SO}(\mathbb{R}^3)$ is non-Abelian.

Parametrizations of $\text{SO}(\mathbb{R}^3)$. Since the abstract rotation group $\text{SO}(\mathbb{R}^3)$ is inaccessible to practical computations, we study some of its *parametrizations*, i. e., continuous, surjective mappings from a parameter space P onto $\text{SO}(\mathbb{R}^3)$. Usually, $P \subseteq \mathbb{R}^n$ for some $n \geq 3$. Desirable properties of such a parametrization $\Pi : P \rightarrow \text{SO}(\mathbb{R}^3)$ are

1. Uniqueness: Π should be bijective.
2. Compatibility with group structure of $\text{SO}(\mathbb{R}^3)$: multiplication and inversion in $\text{SO}(\mathbb{R}^3)$ as well as the action of $\text{SO}(\mathbb{R}^3)$ on \mathbb{R}^3 should be easily expressible in terms of the respective parameter vectors,
3. Continuity of inverse: the inverse image of a closed path in $\text{SO}(\mathbb{R}^3)$ under Π should be a closed path in P .

Especially the last property is often violated. The discontinuity of the parametrization's inverse can then lead to a situation where very similar rotations are encoded by very different parameter vectors, thus precluding a meaningful comparison of rotations by their parameter vectors.

The remainder of this chapter is structured as follows. Starting from the well-known matrix representation in Sect. A.1, we introduce the axis/angle (Sect. A.2), quaternion (Sect. A.3), and exponential parametrizations (Sect. A.4), which are closely related to each other. We then discuss different types of Euler angle parametrizations in Sect. A.5. Each of these parametrizations has its advantages and disadvantages, which are summarized in Tab. A.1. Finally, Sect. A.6 provides a collection of conversion formulas between Euler angles, quaternions, and rotation matrices.

A.1 Rotation Matrices

The most elementary parametrization of the rotation group is obtained by fixing a right-handed orthonormal basis of \mathbb{R}^3 . Then, any $F \in \text{SO}(\mathbb{R}^3)$ can be uniquely represented as an element of the matrix group

$$\text{SO}_3 = \{R \in \text{GL}_3(\mathbb{R}) \mid R^\top = R^{-1}, \det R = 1\}, \quad (\text{A.2})$$

as we have implicitly exploited in Sect. 1.3. The condition $R^\top = R^{-1} \Leftrightarrow RR^\top = E_3$ implies that the row and the column vectors of a matrix $R \in \text{SO}_3$ form an orthonormal basis, and the condition $\det R = 1$ implies that the orientation of this basis is right-handed. Since the columns of any matrix A are the image of the chosen basis under the action of the endomorphism represented by A , these conditions are a restatement of the property that rotations map right-handed orthonormal bases into right-handed orthonormal bases. These facts make it easy to visualize the action of a rotation matrix on \mathbb{R}^3 , cf. Fig. 1.5.

The action of a rotation matrix becomes even clearer by the following considerations. It can be shown that every $R \in \text{SO}_3$ has the eigenvalue 1 with a corresponding one-dimensional R -invariant subspace, the *axis of rotation*. The other two eigenvalues are complex conjugates of the form $e^{\pm i\alpha}$, $\alpha \in (-\pi, \pi]$. It turns out that there is a corresponding two-dimensional R -invariant subspace, the *plane of rotation*, which is orthogonal to the axis of rotation, and that R describes a two-dimensional rotation by an angle of α within the plane of rotation. This implies that an axis of rotation and an angle describing the amount of rotation about this axis can be found for every $R \in \text{SO}_3$. The resulting axis/angle parametrization will be set aside until Sect. A.2.

Performing a change of basis by a 3×3 change matrix C , the columns of which form an orthonormal basis composed of the axis of rotation and an arbitrary orthonormal basis of the

	matrix	axis/angle	quaternion	exp	Euler
redundancy / storage requirements	- -	+	+	++	++
uniqueness of representation	++	o	+	+	- -
overhead for handling parametrization	-	o	+	-	-
composing rotations	+	o	++	o	-
computing action of rotation on vector	+	o	+	o	-
inverting rotations	++	++	+	++	o
constraining rotations	o	+	+	+	++
parameter estimation	o	o	+	++	-
differentiation, integration, optimization	-	+	-	++	o
interpolation properties	- -	-	++	o	- -
global numeric stability	++	++	++	o	- -
comparing rotations by parameters	o	+	++	+	- -
geometric interpretation	+	++	+	+	++
ease of use, manually defining rotations	o	+	-	+	++

Table A.1. Qualitative comparison of five different parametrizations of the rotation group $\text{SO}(\mathbb{R}^3)$. Depending on the respective property, the symbol “+” stands for easy/good, the symbol “-” for difficult/bad, and the symbol “o” for pros and cons/neutral/not applicable.

plane of rotation, the matrix representation of R becomes

$$R' = CRC^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}, \quad (\text{A.3})$$

describing a rotation about the x axis by an angle of α according to the right-hand rule¹. Thus, every 3×3 rotation matrix is similar to a matrix of the form (A.3). Such *basic rotations* about the coordinate axes will play an important role in the Euler angle parametrization, see Sect. A.5.

Just like the isomorphic $\text{SO}(\mathbb{R}^3)$, the matrix group SO_3 is a three-dimensional Lie group. Informally, it is straightforward to imagine why the dimension of SO_3 must be three: there are nine free entries in an arbitrary real 3×3 matrix $R = (r_1, r_2, r_3)$ with $r_i \in \mathbb{R}^3$, $i = 1, 2, 3$. These nine entries are restricted by three nonlinear length conditions ($|r_i| = 1$) and three nonlinear orthogonality conditions ($\langle r_i, r_j \rangle = \langle r_j, t_i \rangle = 0$ for $i \neq j$). It is noted in [Gra98] that the restriction $\det R = 1$ is subtly dependent on the previous six conditions, so it does not count as an additional constraint. Thus, there are three degrees of freedom left, and six of the nine parameters of the matrix representation are redundant. In the following, the group $\text{SO}(\mathbb{R}^3)$ and the matrix group SO_3 are identified with each other.

Discussion. Rotation matrices are the standard choice when it comes to an algebraic treatment of rotations. But even in practical implementations, rotation matrices are widely used in spite of their redundancy. Often, 3×3 rotation matrices appear in a natural way as building blocks of 4×4 homogeneous transformation matrices, which describe the more general class of 3D projective mappings.

¹Pointing the thumb of the right hand in the direction of the axis, the curled fingers indicate the direction of rotation.

Composing two rotations amounts to multiplying the corresponding 3×3 rotation matrices, requiring 27 scalar multiplications and 18 scalar additions if carried out with naive matrix multiplication. The action of a rotation on a vector can be computed by matrix-vector multiplication, which requires 9 multiplications and 6 additions. Inverse rotations are obtained by simply transposing the rotation matrix.

The problem of estimating a 3D rotation from observed point data is similar to the problem of optimally aligning 3D point clouds w. r. t. 2D rigid motions as described in App. C. Given two sets of *observations*, $\{p_1, \dots, p_K\} \subset \mathbb{R}^3$ and $\{p'_1, \dots, p'_K\} \subset \mathbb{R}^3$ with $p_i = Rp'_i + \varepsilon_i$ for an unknown rotation matrix R and noise vectors ε_i , the nine parameters of R can be estimated in the following way. Writing the p_i and p'_i as the columns of the $3 \times K$ matrices P and P' , respectively, we can formulate the problem as $\|P - RP'\|^2 \rightarrow \min$ for some suitable matrix norm $\|\cdot\|$. Up to certain degenerate cases, Arun et al. [AHB87] obtain the least-squares solution from the singular value decomposition $PP'^T = UDV^T$, where $U, V \in O_3$ and D is a 3×3 diagonal matrix, yielding the estimate $\hat{R} = VU^T \in O_3 \subset SO_3$. Umeyama [Ume91] gives a simple extension to their algorithm that is guaranteed to provide the optimal solution in SO_3 even in degenerate cases.

Differentiation, integration, optimization of functions depending on a rotation, and interpolation of rotations are difficult with rotation matrices since these operations lead to or involve matrices that are not elements of SO_3 . This, in turn, necessitates additional non-linear constraints or, in numerical applications, frequent re-orthonormalization, which introduces numerical errors, see [Gra98].

A.2 Axis/Angle

Any 3D rotation can be characterized by a directed *axis of rotation* Rv for $v \in \mathbb{R}^3$ and an *angle of rotation*, $\alpha \in (-\pi, \pi]$. This allows us to define an axis/angle parametrization of SO_3 , which maps pairs of unit-length vectors and angles to SO_3 ,

$$R: S^2 \times S^1 \rightarrow SO_3. \quad (\text{A.4})$$

We will denote the rotation about the axis v by an angle of α by $R_v(\alpha)$. Of course, this parametrization is not unique, since there are infinitely many parameters describing the identity (v arbitrary, $\alpha \in 2\pi\mathbb{Z}$). Here, one has to choose a single representative axis to represent the identity, for example $v = (1, 0, 0)^T$. Also, for any axis of rotation $v \in S^2$, the negative, $-v$, describes the same rotation if the angle of rotation is negated as well. It is therefore sufficient to regard, for example, only the upper hemisphere of S^2 together with one half circle of the equator, one endpoint of which is removed.

The following theorem is due to the French mathematician Benjamin Olinde Rodrigues (1794–1851) and provides an explicit conversion from axis/angle parameters to rotation matrices, see [Fau01, Rod40].

Theorem A.1 (Rodrigues rotation formula) *Let $v \in S^2$ describe an axis of rotation and let $\alpha \in (-\pi, \pi]$ be an angle of rotation. Then*

$$R_v(\alpha) = D_v + \cos \alpha (I_3 - D_v) + \sin \alpha S_v, \quad (\text{A.5})$$

where D_v is the dyadic product vv^\top , I_3 is the 3×3 identity matrix, and S_v is the screw-symmetric cross product matrix for v , defined by

$$S_v := \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \quad (\text{A.6})$$

Proof: Applying the rotation about v to a vector $p \in \mathbb{R}^3$, we obtain the vector $p' = R_v(\alpha)p$. Now our goal is to find an expression for p' . We start by splitting the vector p into a component p_{\parallel} that is parallel to v and a component p_{\perp} that is perpendicular to v , hence $p = p_{\parallel} + p_{\perp}$. The parallel component p_{\parallel} is the orthogonal projection of p onto v , so $p_{\parallel} = \langle v, p \rangle v = v^\top p v = vv^\top p = D_v p$. Therefore, the perpendicular component is given by $p_{\perp} = p - D_v p = (I_3 - D_v)p$. The rotation $R_v(\alpha)$ leaves p_{\parallel} unchanged, since p_{\parallel} is a scalar multiple of v , while the component p_{\perp} is rotated by a 2D rotation within the orthogonal complement $E := (\text{span}\{v\})^\perp$ of v . We express this 2D rotation in a right-handed orthogonal basis for E , which is given by $(p_{\perp}, v \times p) = ((I_3 - D_v)p, S_v p)$. The vector $v \times p$ is perpendicular to both v and p_{\perp} , and it has the same length as p_{\perp} since $\|v \times p\| = \|v \times (p_{\parallel} + p_{\perp})\| = \|v \times p_{\perp}\| = |\sin \frac{\pi}{2}| \|p_{\perp}\| = \|p_{\perp}\|$. This gives us the following expression for the rotated vector p' :

$$\begin{aligned} p' &= p_{\parallel} + \cos \alpha p_{\perp} + \sin \alpha (v \times p) \\ &= D_v p + \cos \alpha (I_3 - D_v)p + \sin \alpha S_v p \\ &= (D_v + \cos \alpha (I_3 - D_v) + \sin \alpha S_v) p, \end{aligned} \quad (\text{A.7})$$

which concludes the proof. \square

Discussion. The axis/angle representation requires four parameters, where the redundant DOF is the unit length constraint on the three axis parameters. The parametrization has an intuitive geometric interpretation. Inverting a rotation given as an axis/angle pair can be done by negating either the axis or the angle. Composing two rotations about the same axis is trivial, but there is no easy way of composing rotations about different axes in terms of their axis/angle representation. Applying an axis/angle rotation to a vector can either be done by evaluating the Rodrigues formula or by first converting the axis/angle values to a unit quaternion, see Sect. A.3. Many physically-based applications involving derivatives of rotations for kinematics and dynamics make use of the axis/angle parametrization.

A.3 Quaternions

Similar to the axis/angle representation, the concept of quaternions allows for a compact parametrization of 3D rotations by four parameters. From a differential geometric point of view, the set of unit quaternions has a structure that is compatible with SO_3 and is therefore particularly well-suited for interpolation of rotations. Therefore, quaternions have become an indispensable tool for the analysis and synthesis of human motions. In this section, we first summarize some basic mathematical properties about quaternions and then describe their relation to rotations. For further details, we refer to the literature such as [Por95, Sho85].

Quaternions have been introduced by Sir William Rowan Hamilton in 1843 as a non-commutative extension of the complex numbers. Denoting the standard basis of \mathbb{R}^4 by the

symbols $1, i, j, k$, a *quaternion* $q \in \mathbb{R}^4$ can be written uniquely as $q = w + xi + yj + zk$ for suitable $w, x, y, z \in \mathbb{R}$. The part $\operatorname{Re}(q) := w \in \mathbb{R}$ —a scalar—is called the *real part* of q , whereas the part $\operatorname{Im}(q) := (x, y, z)^\top \in \mathbb{R}^3$ —a vector—is called the *imaginary part* of q . A quaternion q is called *pure* if $w = 0$; the set of pure quaternions will be denoted as \mathbb{H}^0 . In the following, we will also identify a vector $p = (x, y, z)^\top \in \mathbb{R}^3$ with a pure quaternion $\tilde{p} := xi + yj + zk \in \mathbb{H}^0$. For two quaternions, $q_1 = w_1 + x_1i + y_1j + z_1k$ and $q_2 = w_2 + x_2i + y_2j + z_2k$, one defines the product quaternion $q_1 \cdot q_2$ by

$$\begin{aligned} q_1 \cdot q_2 &:= w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ &\quad + (w_1x_2 + w_2x_1 + y_1z_2 - z_1y_2)i \\ &\quad + (w_1y_2 + w_2y_1 + z_1x_2 - x_1z_2)j \\ &\quad + (w_1z_2 + w_2z_1 + x_1y_2 - y_1x_2)k. \end{aligned} \tag{A.8}$$

Now, it is an easy but tedious exercise to show that this indeed defines an associative and distributive multiplication on \mathbb{R}^4 with neutral element $e = 1$. Usually, one simply writes q_1q_2 for $q_1 \cdot q_2$. For a quaternion $q = w + xi + yj + zk$, the *conjugate quaternion* is defined as $\bar{q} := w - xi - yj - zk$ and the *norm* of q is defined as $\|q\| := \sqrt{w^2 + x^2 + y^2 + z^2}$. A straightforward computation shows that $q^{-1} := \bar{q}/\|q\|^2$ defines a right and left inverse in the case $q \neq 0$, i. e., $qq^{-1} = q^{-1}q = 1$. Note that the multiplication is not commutative—for example, one has $ij = -ji$. Altogether, we have seen that \mathbb{R}^4 equipped with vector addition and the multiplication defined by (A.8) forms a skew field, which is often denoted by \mathbb{H} in honor of Hamilton.

For the imaginary numbers $i, j, k \in \mathbb{H}$, the multiplication induces the following famous relations:

$$i^2 = j^2 = k^2 = -1, \quad ij = k, \quad jk = i, \quad ki = j. \tag{A.9}$$

Actually, the multiplication in (A.8) is uniquely determined by these relations if one requires the multiplication to be associative and distributive. Note that one also has $ij = -ji$, $jk = -kj$, and $ki = -ik$. To obtain more readable formulas, one often writes a quaternion $q \in \mathbb{H}$ as a tuple (s, v) with $s = \operatorname{Re}(q)$ and $v = \operatorname{Im}(q)$. Then, the following rules can be formulated for quaternions $q_1 = (s_1, v_1)$ and $q_2 = (s_2, v_2)$:

$$q_1q_2 = (s_1s_2 - \langle v_1, v_2 \rangle, s_1v_2 + s_2v_1 + v_1 \times v_2) \tag{A.10}$$

$$\|q\|^2 = q\bar{q} = s^2 + \langle v, v \rangle \tag{A.11}$$

$$q^{-1} = \frac{(s, -v)}{\|q\|^2}. \tag{A.12}$$

A quaternion of norm one is also referred to as a *unit quaternion*. The set of unit quaternions forms a hypersphere $S^3 \subset \mathbb{H}$. It is not difficult to see that any unit quaternion $q = w + xi + yj + zk$ can be written as $q = (\cos \alpha, u \sin \alpha)$ for a suitable angle $\alpha \in (-\pi, \pi]$ and a unit vector $u \in \mathbb{R}^3$: setting $\cos^2 \alpha := w^2$, we may conclude that $x^2 + y^2 + z^2 = \sin^2 \alpha$ due to the relation $\sin^2 \alpha + \cos^2 \alpha = 1$. Hence $q = (\cos \alpha, u \sin \alpha)$ with the unit vector $u := (x, y, z)^\top / \|(x, y, z)^\top\|$. The following results show that quaternions can be used to describe rotations of \mathbb{R}^3 .

Theorem A.2

(a) For every non-zero quaternion $q \in \mathbb{H}^* := \mathbb{H} \setminus \{0\}$, the map ρ_q defined by

$$\rho_q(p) := q\tilde{p}q^{-1} \tag{A.13}$$

for $p \in \mathbb{R}^3$, is a rotation of \mathbb{R}^3 .

(b) Writing $q \in \mathbb{H}^*$ in the form $q = (\cos \alpha, u \sin \alpha)$ for a suitable angle $\alpha \in (-\pi, \pi]$ and a unit vector $w \in \mathbb{R}^3$, ρ_q describes a rotation about the axis u by an angle of 2α .

(c) The map

$$\rho : \mathbb{H}^* \rightarrow \text{SO}_3, \quad q \mapsto \rho_q \quad (\text{A.14})$$

is a surjective group homomorphism from the multiplicative group \mathbb{H}^* onto the group SO_3 with kernel $\mathbb{R}^* := \mathbb{R} \setminus \{0\}$. The restriction of ρ to $S^3 \subset \mathbb{H}^*$ is also a surjective group homomorphism with kernel $\{\pm 1\}$.

Proof: Since ρ is homogeneous, i. e., $\rho_q = \rho_{\lambda q}$ for all $\lambda \in \mathbb{R}^*$ and $q \in S^3$, we may restrict our considerations to unit quaternions q , which we assume to be given as $q = (s, v) = (\cos \alpha, u \sin \alpha)$ with $\alpha \in (-\pi, \pi]$, $u \in S^2$. We can then evaluate $\rho_q(p)$ of (A.13), yielding the quaternion \tilde{p}' :

$$\begin{aligned} \tilde{p}' &= q\tilde{p}\bar{q} \\ &= (s, v)(0, p)(s, -v) \\ &= (-\langle v, p \rangle, sp + v \times p)(s, -v) \\ &= (-s\langle v, p \rangle - \langle sp + v \times p, -v \rangle, v\langle v, p \rangle + s(sp + v \times p) + (sp + v \times p) \times (-v)) \\ &= (0, 2v\langle v, p \rangle + 2s(v \times p) + (s^2 - \|v\|^2)p), \end{aligned} \quad (\text{A.15})$$

where the Grassmann identity $a \times (b \times c) = b\langle a, c \rangle - c\langle a, b \rangle$ was used in the last step. With the notation of Sect. A.2, this can be written as

$$p' = \underbrace{(2D_v + 2sS_v + (s^2 - \|v\|^2)I_3)}_{R(q)} p. \quad (\text{A.16})$$

Substituting $s = \cos \alpha$, $v = u \sin \alpha$, and $\|v\|^2 = \sin^2 \alpha$, we obtain

$$\begin{aligned} R(q) &= 2D_v + 2sS_v + (s^2 - \|v\|^2)I_3 \\ &= 2\sin^2 \alpha D_u + 2\cos \alpha \sin \alpha S_u + (\cos^2 \alpha - \sin^2 \alpha)I_3 \\ &= (1 - \cos(2\alpha))D_u + \sin(2\alpha)S_u + \cos(2\alpha)I_3 \\ &= D_u + \cos(2\alpha)(I_3 - D_u) + \sin(2\alpha)S_u, \end{aligned} \quad (\text{A.17})$$

where we used the double-angle formulas $\cos(2\alpha) = 1 - 2\sin^2 \alpha = \cos^2 \alpha - \sin^2 \alpha$ and $\sin(2\alpha) = 2\sin \alpha \cos \alpha$. The result (A.17) is exactly the Rodrigues rotation formula (A.5), describing a rotation about the axis u by an angle of 2α . This proves (a) and (b).

To prove (c), one easily checks that the homomorphism properties $\rho_{q_1 q_2} = \rho_{q_1} \circ \rho_{q_2}$ and $\rho_{1_{\mathbb{H}}} = 1_{\text{SO}_3}$ hold. Furthermore, we found that ρ yields rotations according to the Rodrigues rotation formula, so the surjectivity of ρ follows from the surjectivity of the axis/angle parametrization, since one can define a surjective mapping $(\cos \alpha, u \sin \alpha) \mapsto (u, 2\alpha)$ from S^3 to the axis/angle parameter space $S^2 \times S^1$, see Sect. A.2. The kernel properties of ρ follow from $\rho_q = 1_{\text{SO}_3} \Leftrightarrow \forall p \in \mathbb{R}^3 : \rho_q(p) = p \Leftrightarrow \forall p \in \mathbb{R}^3 : q\tilde{p} = \tilde{p}q$, i. e., q is in the kernel of ρ iff it commutes with all pure quaternions. The set of such quaternions q can be shown to be \mathbb{R} . The claimed properties for the restriction of ρ to S^3 follow from the homogeneity of ρ . \square

Example A.3 As an illustration, we consider the rotation of the vector $p = (1, 0, 1)^\top \in \mathbb{R}^3$ about the y axis by $2\alpha = \pi$. Regarded as a quaternion, p reads as $\tilde{p} = (0, 1, 0, 1)^\top$. By Thm. A.2 (b), the rotation is given by the quaternion $q = (\cos \frac{\pi}{2}, \sin \frac{\pi}{2}(0, 1, 0)^\top)^\top = (0, 0, 1, 0)^\top$, the inverse of which is $q^{-1} = \bar{q} = (0, 0, -1, 0)^\top$. Then

$$\begin{aligned} \rho_q(p) = q\tilde{p}\bar{q} &= (0, 0, 1, 0)^\top (0, 1, 0, 1)^\top (0, 0, -1, 0)^\top \\ &= (0, 1, 0, -1)^\top (0, 0, -1, 0)^\top = (0, -1, 0, -1)^\top. \end{aligned} \quad (\text{A.18})$$

The map ρ describes a parametrization of rotations using S^3 as the parameter space. Identifying antipodal points on S^3 as suggested by Thm. A.2 (c), one obtains the space $\mathbb{RP}(3) := S^3/\{\pm 1\}$ known as real three-dimensional *projective space*. Then Thm. A.2 (c) implies that ρ induces a bijective map $\mathbb{RP}(3) \rightarrow \text{SO}_3$. Even more, one can show that this map is continuous with a continuous inverse map. From a differential geometric point of view, ρ exhibits the spherical geometry of SO_3 . The practical consequence of this parametrization is that small changes of a rotation in SO_3 also result in small changes of the corresponding quaternion—the parametrization of SO_3 based on quaternions is free from gimbal lock, cf. Sect. A.5.4. Furthermore, any smooth path in SO_3 corresponds to a smooth path in $\mathbb{RP}(3)$, and vice versa.

Spherical Linear Interpolation. In animation systems, one often works with more or less sparse sequences of *keyframes*. Each keyframe describes the state of an animated object, a virtual camera, or a virtual light source. Then, the goal is to derive a smooth animation from such a sequence of keyframes by using interpolation techniques for 3D positions as well as 3D rotations. Due to the reasons discussed above, quaternions are ideally suited for interpolating rotations.

There are many possible ways of interpolating between two unit quaternions $q_0, q_1 \in S^3$. For example, one could simply form convex combinations in \mathbb{R}^4 . Evaluating the convex combination $q(t) = tq_0 + (1-t)q_1$ for a parameter $t \in [0, 1]$, one would obtain a straight line of quaternions connecting q_0 and q_1 that cuts through the volume enclosed by the unit sphere S^3 . Identifying the parameter t with time, one finds that such an interpolation leads to rotations with non-constant angular velocity: while departing from the orientation described by q_0 , the rotation accelerates and then decelerates while approaching q_1 , see [Sho85].

Here, the solution is to interpolate within S^3 . Then, the quaternion path runs along a unit radius great circle arc on S^3 , resulting in a smooth interpolation between two orientations by means of a time-dependent rotation of constant angular velocity. There are two different approaches to computing the resulting *spherical linear interpolation* (slerp) between q_0 and q_1 , which both yield the same result, see [Sho85]. The first approach uses the group structure of S^3 and yields

$$\text{slerp}(t; q_0, q_1) = q_0(q_0^{-1}q_1)^t \quad (\text{A.19})$$

for $t \in [0, 1]$, where the quaternion power q^t can be evaluated by the quaternionic logarithm and the quaternionic exponential as $\exp(t \log q)$, see Sect. A.4 and [Gra98, LS02]. The second approach is geometrically motivated and gives

$$\text{slerp}(t; q_0, q_1) = \frac{\sin(1-t)\alpha}{\sin \alpha} q_0 + \frac{\sin t\alpha}{\sin \alpha} q_1, \quad (\text{A.20})$$

where $\langle q_0, q_1 \rangle = \cos \alpha$. For further details, we refer to [Sho85].

Enforcing Path Continuity on S^3 . In real-world mocap data, angle trajectories are typically described by Euler angles, see Sect. A.5. Usually, the first step in any system that processes mocap data is to convert the human-readable Euler angles into some representation that is well-suited for practical computations, such as unit quaternions. The formulas given in Sect. A.6.2 provide this conversion. Since we assume human motion to be continuous in nature, we would like this continuity to be reflected in the angle trajectories. Of course, the strict notion of continuity does not apply to our sampled data. However, unit quaternions bear the risk of representing even continuous-time, unsampled motions as discontinuous trajectories, since antipodal points on S^3 are identified with each other. In effect, a continuous path on S^3 is free to jump to $-q$ at any point q along the path while still representing the same trajectory in SO_3 . Such an effect may also occur after our Euler-to-quaternion conversion. The following simple strategy, see [Gra98, LS02], can be applied to suppress such “flips” in a given sequence of unit quaternions $q_1, \dots, q_T \in S^3$: construct a modified sequence $q'_1, \dots, q'_T \in S^3$, where successive quaternions are guaranteed to lie in the same hemisphere:

$$\begin{aligned} q'_1 &:= q_1 \\ q'_{t+1} &:= \begin{cases} q_{t+1}, & \text{if } \langle q_{t+1}, q_t \rangle \geq 0 \\ -q_{t+1}, & \text{otherwise.} \end{cases} \end{aligned} \quad (\text{A.21})$$

Quaternion-to-Matrix Conversion. Given a unit quaternion $q = (w, x, y, z)^\top = (s, v)$, we want to express the action of the “sandwich product” ρ_q on a vector $p \in \mathbb{R}^3$ in matrix form. We know from (A.16) that $p' = R(q)p$ with a rotation matrix

$$\begin{aligned} R(q) &= 2D_v + 2sS_v + (s^2 - \|v\|^2)I_3 \\ &= \begin{pmatrix} 2x^2 & 2xy & 2xz \\ 2xy & 2y^2 & 2yz \\ 2xz & 2yz & 2z^2 \end{pmatrix} + \begin{pmatrix} 0 & -2wz & 2wy \\ 2wz & 0 & -2wx \\ -2wy & 2wx & 0 \end{pmatrix} + (w^2 - x^2 - y^2 - z^2)I_3 \\ &= \begin{pmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & w^2 - x^2 - y^2 + z^2 \end{pmatrix}. \end{aligned} \quad (\text{A.22})$$

Exploiting the unit length condition $w^2 + x^2 + y^2 + z^2 = 1$, we can achieve a slight simplification of the diagonal entries:

$$R(q) = \begin{pmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{pmatrix}. \quad (\text{A.23})$$

These formulas are due to A. Cayley (1845). For an easy way of converting from rotation matrices to quaternions, we refer to [Sho85].

Caveat: In the literature, especially in references from the graphics community such as [Sho85], one also finds a transposed version of $R(q)$. The reason is that the authors represent vectors in \mathbb{R}^3 as row vectors instead of column vectors.

Discussion. Quaternions are a compact representation of rotations using four real parameters with one constraint (normalization of unit quaternion). The composition and inversion of rotations correspond to multiplication and inversion in \mathbb{H} , yielding easy and direct formulas.

The equivalent to re-orthonormalization as required for rotation matrices during numerical computations is a simple re-normalization for unit quaternions. This operation is much simpler to compute and does not change the rotation that is described by the quaternion. Working with unit quaternions has the computational advantage that the inverse required in (A.13) amounts to simple conjugation. Evaluating the map ρ_q for a unit quaternion $q \in S^3$ on a vector $p \in \mathbb{R}^3$ requires roughly as many multiplications as matrix-vector multiplication. However, multiplying two quaternions requires only 16 scalar multiplications and 10 scalar additions, cf. (A.10). Constraining rotations to a specified range using quaternions is possible by the technique due to Liu and Prakash [LP03]. One of the main applications of quaternions is spherical linear interpolation (slerp) for the purpose of computing meaningful intermediate rotations. As a drawback, quaternions are not as intuitive as other parametrizations and are therefore typically not found in the front end of applications involving 3D rotations.

A.4 Exponential Map

The Rodrigues rotation formula (A.5) was originally introduced as a shortcut for evaluating the matrix exponential $\exp(S) = \sum_{k=0}^{\infty} \frac{S^k}{k!}$ for skew-symmetric 3×3 matrices S , see [Fau01]. It is easy to see that the matrix exponential of S must be a rotation matrix: from $\exp(S)^\top = \exp(S^\top) = \exp(-S)$, we obtain $\exp(S) \exp(S)^\top = \exp(S - S) = I_3$, since S is a normal matrix and therefore commutes with its transpose, and since commuting matrices A, B satisfy the familiar relation $\exp(A) \exp(B) = \exp(A + B)$. Recall that the Rodrigues rotation formula involved the skew-symmetric matrix S_v for a unit vector $v \in \mathbb{R}^3$. It turns out that $\exp(\alpha S_v) = R_v(\alpha)$ for an angle α . Since the space of skew-symmetric 3×3 matrices is isomorphic to \mathbb{R}^3 via S_v , one can also define an exponential map $\mathbb{R}^3 \rightarrow \text{SO}_3$ that directly maps vectors $\alpha v \in \mathbb{R}^3$ to $\exp(\alpha S_v) = R_v(\alpha)$. Now, such a construction still works if one replaces SO_3 by the set of unit quaternions, S^3 , see [Gra98]. To this end, we identify \mathbb{R}^3 with the set of pure quaternions, \mathbb{H}^0 , and define the *quaternionic exponential map* from \mathbb{H}^0 to S^3 as follows:

$$\exp(0, v) := \begin{cases} (1, 0, 0, 0)^\top & \text{if } v = (0, 0, 0)^\top \\ \left(\cos\left(\frac{\|v\|}{2}\right), \frac{\sin\left(\frac{\|v\|}{2}\right)}{\|v\|} v \right) & \text{otherwise,} \end{cases} \quad (\text{A.24})$$

where $v \in \mathbb{R}^3$. We know from Sect. A.3 that the quaternion $\exp(0, v)$ describes a rotation about the axis v by an angle of $\alpha := \|v\|$. The numerical instability as α approaches zero can be handled by using the identity

$$\frac{\sin\left(\frac{\alpha}{2}\right)}{\alpha} = \frac{1}{2} \text{sinc}\left(\frac{\alpha}{2}\right), \text{ where } \text{sinc}(x) := \begin{cases} \frac{\sin x}{x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases} \quad (\text{A.25})$$

since the sinc function is well-defined and continuous at the origin. In terms of Lie theory, the quaternionic exponential maps the tangent space of S^3 at the identity, $T_1 S^3 \cong \mathbb{H}^0$, to S^3 while preserving distances and angles, see [BF01, LS02]. Exploiting the group structure of S^3 , we can define the exponential map for the tangent space $T_{q_0} S^3$ at an arbitrary quaternion $q_0 \in S^3$ by means of a change of coordinates:

$$\exp_{q_0}(q) := q_0 \exp(q_0^{-1} q), \quad (\text{A.26})$$

where $q \in T_{q_0} S^3$.

The resulting parametrization is not unique: adding multiples of 2π to the length of a parameter vector $v \in \mathbb{R}^3$ will yield the same rotation. However, by exploiting that a rotation about v by α is equivalent to a rotation about $-v$ by $2\pi - \alpha$, one can restrict the parameter space to the open ball $B := \{v \in \mathbb{R}^3 \mid \|v\| < \pi\}$ by dynamic reparametrization, thus yielding a one-to-one mapping from B to $S^3 \setminus \{(-1, 0, 0, 0)^\top\}$, see also [Gra98]. This allows us to formulate a unique inverse mapping for the quaternionic exponential. Given a unit quaternion $q = (s, w) \in S^3 \setminus \{(-1, 0, 0, 0)^\top\}$, we define the *quaternionic logarithm* of q as

$$\log q := \begin{cases} \left(0, \frac{2 \arccos s}{\|w\|} w\right) & \text{if } q \neq (1, 0, 0, 0)^\top \\ (0, 0, 0, 0)^\top & \text{otherwise .} \end{cases} \quad (\text{A.27})$$

Since the exponential map gives us $\|w\| = \frac{\sin \frac{\alpha}{2}}{\alpha}$, where $\alpha = 2 \arccos s$, we can once again use the sinc function to handle the numerical instability for α approaching zero: $\|w\| = \frac{1}{2} \operatorname{sinc}\left(\frac{\alpha}{2}\right) \xrightarrow{\alpha \rightarrow 0} \frac{1}{2}$. The quaternionic logarithm maps S^3 to its tangent space at the identity, $T_1 S^3$. In analogy to (A.26), we define the logarithm map for the tangent space $T_{q_0} S^3$ at an arbitrary quaternion $q_0 \in S^3$:

$$\log_{q_0}(q) := q_0 \log(q_0^{-1} q), \quad (\text{A.28})$$

where $q \in S^3 \setminus \{-q_0\}$. One can check that $\exp_{q_0} \circ \log_{q_0} = \log_{q_0} \circ \exp_{q_0} = \operatorname{id}_{\mathbb{H}}$ for all $q_0 \in S^3$.

Discussion. The main advantage of the exponential map parametrization over the representations that have been discussed so far is that it only requires three parameters. Computing the inverse rotation is simple, while composing rotations in terms of the parameter vector is only possible if the axes are the same. To compute the action of a rotation on a vector, one first has to evaluate the corresponding quaternion q and then use the mapping ρ_q . The reason why the exponential map has become popular in computer animation, mainly for inverse kinematics and dynamics, are the good properties of the parametrization's derivatives. The exponential map is also used for interpolation and filtering of quaternion sequences, see App. B.

A.5 Euler Angles

Leonhard Euler (1707-1783) proved that any 3D rotation can be expressed as a sequence of three basic rotations about the coordinate axes. The three angles of rotation are referred to as *Euler angles*, occasionally also as *Tait-Bryan* or *Cardan angles*, see also [Zat98]. In order to use Euler angles as a parametrization for SO_3 , one has to make some choices:

- which axes to rotate about,
- in which order to rotate,
- whether to express rotations relative to a fixed coordinate frame or relative to a coordinate frame that moves along with the basic rotations.

As usual, we will refer to the three axes of our right-handed coordinate system as x , y , and z . For the time being, we will adhere to the following convention:

1. The first rotation, $R_x(\alpha_1)$, rotates by an angle of α_1 about the x axis of the coordinate system $(0; x, y, z)$.

2. The second rotation, $R_y(\alpha_2)$, rotates by an angle of α_2 about the y axis of the coordinate system $(0; x, y, z)$.
3. The third rotation, $R_z(\alpha_3)$, rotates by an angle of α_3 about the z axis of the coordinate system $(0; x, y, z)$.

We will refer to this convention as the \overleftarrow{xyz} Euler convention. The meaning of the arrow will become clear in Sect. A.5.1, when we discuss the difference between fixed and moving reference frames. At this point, we assume a fixed reference frame—the three rotation matrices $R_i \in \text{SO}_3$, $i = 1, 2, 3$, are representations of linear mappings with respect to the same basis. The matrix representing the composition of these linear mappings is then obtained in the usual way, i. e., by multiplying the respective matrices in a right-to-left sequence. Hence, the effect of the above sequence of rotations is described by

$$R_{\overleftarrow{xyz}}(\alpha_1, \alpha_2, \alpha_3) := R_z(\alpha_3)R_y(\alpha_2)R_x(\alpha_1), \quad (\text{A.29})$$

where the three basic rotation matrices are

$$R_x(\alpha_1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha_1 & -\sin \alpha_1 \\ 0 & \sin \alpha_1 & \cos \alpha_1 \end{pmatrix}, \quad (\text{A.30})$$

$$R_y(\alpha_2) = \begin{pmatrix} \cos \alpha_2 & 0 & \sin \alpha_2 \\ 0 & 1 & 0 \\ -\sin \alpha_2 & 0 & \cos \alpha_2 \end{pmatrix}, \quad (\text{A.31})$$

$$R_z(\alpha_3) = \begin{pmatrix} \cos \alpha_3 & -\sin \alpha_3 & 0 \\ \sin \alpha_3 & \cos \alpha_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (\text{A.32})$$

with $\alpha_i \in (-\pi, \pi]$. Introducing the abbreviations $c_i := \cos \alpha_i$ and $s_i := \sin \alpha_i$ for $i = 1, 2, 3$, the final rotation matrix can be worked out to be

$$\begin{aligned} R_{\overleftarrow{xyz}}(\alpha_1, \alpha_2, \alpha_3) &= R_z(\alpha_3)R_y(\alpha_2)R_x(\alpha_1) \\ &= \begin{pmatrix} c_3c_2 & -s_3c_1 + c_3s_2s_1 & s_3s_1 + c_3s_2c_1 \\ s_3c_2 & c_3c_1 + s_3s_2s_1 & -c_3s_1 + s_3s_2c_1 \\ -s_2 & c_2s_1 & c_2c_1 \end{pmatrix} \end{aligned} \quad (\text{A.33})$$

The resulting Euler angle parametrization by can be thought of as a mapping

$$R_{\overleftarrow{xyz}} : S^1 \times S^1 \times S^1 \rightarrow \text{SO}_3. \quad (\text{A.34})$$

As an example, let us consider the case $\alpha_i = \frac{\pi}{2}$ for $i = 1, 2, 3$. To visualize these Euler angles under the \overleftarrow{xyz} convention, place a book on the table in front of you so that the front cover faces upwards and the spine of the book faces you. We assume that the x axis points to the right, the y axis points upwards, and the z axis points straight at you². The first rotation is $R_x(\frac{\pi}{2})$, so rotate the book by 90° about the x axis. The book now sits on its spine, the front cover facing towards you. The second rotation is $R_y(\frac{\pi}{2})$, which rotates the book counter-clockwise by 90° on its spine so the front cover now faces to the right. The third rotation is

²It is helpful to use the thumb, the index finger, and the middle finger of your right hand to imagine the x , y , and z axis, respectively.

$R_z\left(\frac{\pi}{2}\right)$, which once more brings the front cover to the top, with the spine facing to the right. In effect, we have rotated the book by 90° about the y axis. Using matrix notation, this can be seen from the identity

$$R_{\overline{xyz}}\left(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}\right) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} = R_y\left(\frac{\pi}{2}\right). \quad (\text{A.35})$$

Our example also shows that this form of a parametrization is not a bijective mapping onto SO_3 since we have just found at least two triples of Euler angles describing the same rotation under the \overline{xyz} convention: $\left(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}\right)^\top$ and $\left(0, \frac{\pi}{2}, 0\right)^\top$. To obtain a bijective mapping, the parameter space needs to be restricted in a suitable way. Such issues are due to the *gimbal lock* effect, which will be discussed in detail in Sect. A.5.4.

A further point regarding the uniqueness of Euler angles concerns the domain of the angle α_2 . Even though the basic rotation $R_y(\alpha_2)$ is defined for $\alpha_2 \in (-\pi, \pi]$, it turns out that the domain of α_2 can be restricted to $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ in the composite rotation matrix $R_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3)$ because of the symmetry

$$R_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3) = R_{\overline{xyz}}(\pi + \alpha_1, \pi - \alpha_2, \pi + \alpha_3) \quad (\text{A.36})$$

This is due to the following symmetry relations for sines and cosines, which are valid for all $\varphi \in \mathbb{R}$:

$$\begin{aligned} \cos(\pi + \varphi) &= -\cos \varphi & \cos(\pi - \varphi) &= -\cos \varphi \\ \sin(\pi + \varphi) &= -\sin \varphi & \sin(\pi - \varphi) &= +\sin \varphi. \end{aligned} \quad (\text{A.37})$$

Therefore, the transition $(\alpha_1, \alpha_2, \alpha_3)^\top \mapsto (\alpha'_1, \alpha'_2, \alpha'_3)^\top := (\pi + \alpha_1, \pi - \alpha_2, \pi + \alpha_3)^\top$ as described in (A.36) flips the signs of all $c_1, s_1, c_2, c_3,$ and s_3 terms in the matrix $R_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3)$. However, it is easy to verify that these sign changes cancel out due to the special arrangement of sine and cosine products. The term s_2 , which only appears once as an isolated term in the matrix, remains unchanged. Hence, the entire matrix does not change.

The value of the angle α_2 can then be forced into the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ for any given triple of Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top \in (-\pi, \pi]^3$. In case $\alpha_2 \in \left(-\pi, -\frac{\pi}{2}\right) \cup \left(\frac{\pi}{2}, \pi\right]$, we reparameterize to the angles $(\alpha'_1, \alpha'_2, \alpha'_3)^\top := (\pi + \alpha_1, \pi - \alpha_2, \pi + \alpha_3)^\top$, where angle values beyond the interval $(-\pi, \pi]$ are reduced back into this interval modulo 2π . The resulting parameter space for the \overline{xyz} Euler convention is $\mathcal{E}_3 := (-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times (-\pi, \pi]$.

A.5.1 Moving Reference Frame vs. Fixed Reference Frame

It has been mentioned that another possible convention for performing successive rotations is to rotate relative to a moving reference frame. In other words, we imagine a coordinate frame that is attached to the rotating object, and successive rotations take place about the axes of this rotated coordinate frame:

1. The first rotation, $R_x(\alpha_1)$, rotates by an angle of α_1 about the x axis of the coordinate system $(0; x, y, z)$.
2. The second rotation, $R_{y'}(\alpha_2)$, rotates by an angle of α_2 about the y' axis of the rotated coordinate system $(0; x, y', z') := (0; x, R_x(\alpha_1)y, R_x(\alpha_1)z)$.

3. The third rotation, $R_{z''}(\alpha_3)$, rotates by an angle of α_3 about the z'' axis of the twice-rotated coordinate system $(0; x', y', z'') := (0; R_{y'}(\alpha_2)x, R_x(\alpha_1)y, R_{y'}(\alpha_2)R_x(\alpha_1)z)$.

The composite rotation can then be written as

$$R(\alpha_1, \alpha_2, \alpha_3) := R_{z''}(\alpha_3)R_{y'}(\alpha_2)R_x(\alpha_1). \quad (\text{A.38})$$

The rotations $R_{y'}(\alpha_2)$ and $R_{z''}(\alpha_3)$ do not refer to the axes of the world coordinate system $(0; x, y, z)$. To represent these rotations in terms of rotations about the world coordinate axes, we use the following general rule:

Lemma A.4 *Let $v_1, v_2 \in \mathbb{R}^3$, $\|v_1\| = \|v_2\| = 1$, represent two axes of rotation, and let $\alpha \in (-\pi, \pi]$ be an angle of rotation. If the two axes are related by $v_2 = Rv_1$ for a rotation $R \in \text{SO}_3$, then*

$$R_{v_2}(\alpha) = RR_{v_1}(\alpha)R^\top, \quad (\text{A.39})$$

i. e., $R_{v_2}(\alpha)$ results from $R_{v_1}(\alpha)$ by means of a change of basis by R .

Proof: The axis of rotation is the eigenspace of a rotation matrix corresponding to the eigenvalue 1. Using $v_2 = Rv_1 \Leftrightarrow v_1 = R^\top v_2$, it is easy to see that v_2 must span the axis of rotation of $RR_{v_1}(\alpha)R^\top$:

$$RR_{v_1}(\alpha)R^\top v_2 = RR_{v_1}(\alpha)v_1 = Rv_1 = v_2. \quad (\text{A.40})$$

Since both $R_{v_1}(\alpha)$ and $R_{v_2}(\alpha)$ rotate by an angle of α , this concludes the proof. \square

From the above considerations, we know that the y' axis results from the y axis by applying the rotation $R_x(\alpha_1)$, so $y' = R_x(\alpha_1)y$. Then, Lemma A.4 yields

$$R_{y'}(\alpha_2) = R_x(\alpha_1)R_y(\alpha_2)R_x^\top(\alpha_1). \quad (\text{A.41})$$

Similarly, $z'' = R_{y'}(\alpha_2)R_x(\alpha_1)z = R_x(\alpha_1)R_y(\alpha_2)z$ gives us

$$R_{z''}(\alpha_3) = R_x(\alpha_1)R_y(\alpha_2)R_z(\alpha_3)R_y^\top(\alpha_2)R_x^\top(\alpha_1). \quad (\text{A.42})$$

Putting it all together, we obtain the following formula:

$$\begin{aligned} R(\alpha_1, \alpha_2, \alpha_3) &= R_{z''}(\alpha_3)R_{y'}(\alpha_2)R_x(\alpha_1) \\ &= R_x(\alpha_1)R_y(\alpha_2)R_z(\alpha_3)R_y^\top(\alpha_2)R_x^\top(\alpha_1)R_x(\alpha_1)R_y(\alpha_2)R_x^\top(\alpha_1)R_x(\alpha_1) \\ &= R_x(\alpha_1)R_y(\alpha_2)R_z(\alpha_3). \end{aligned} \quad (\text{A.43})$$

Obviously, Euler rotations w. r. t. a moving reference frame can be expressed in the world coordinate system by applying the basic rotations in reversed order. This motivates the following notation:

$$R_{\overrightarrow{xyz}}(\alpha_1, \alpha_2, \alpha_3) := R_x(\alpha_1)R_y(\alpha_2)R_z(\alpha_3), \quad (\text{A.44})$$

where the left-to-right arrow above the ' xyz ' indicates that the corresponding basic rotations are multiplied from left to right. Hence, we use the left-to-right arrow to indicate that the basic rotations refer to a moving reference frame.

A.5.2 Further Euler Conventions

So far, we have always used xyz conventions, where we first rotate about the x axis, then about the y axis, and then about the z axis. Of course, it is possible to define other Euler conventions with a different order of performing the basic rotations. In general, each of the six permutations $\sigma : \{1, 2, 3\} \rightarrow \{x, y, z\}$ of the axes x , y , and z gives rise to two Euler conventions by the following definition:

$$R_{\overrightarrow{\sigma}}(\alpha_1, \alpha_2, \alpha_3) := R_{\sigma(1)}(\alpha_1)R_{\sigma(2)}(\alpha_2)R_{\sigma(3)}(\alpha_3) \quad (\text{A.45})$$

$$R_{\overleftarrow{\sigma}}(\alpha_1, \alpha_2, \alpha_3) := R_{\sigma(3)}(\alpha_3)R_{\sigma(2)}(\alpha_2)R_{\sigma(1)}(\alpha_1), \quad (\text{A.46})$$

These six permutations are $\Sigma_3 := \{xyz, xzy, yxz, yzx, zxy, zyx\}$. We call the permutation σ the *rotation order* of an Euler convention, while the left-to-right or right-to-left arrows indicate the *multiplication order*. Of course, there is a close relationship between the two multiplication orders. Comparing (A.45) and (A.46), we can derive the following obvious conversion formulas:

$$R_{\overrightarrow{\sigma}}(\alpha_1, \alpha_2, \alpha_3) = R_{\overleftarrow{r(\sigma)}}(\alpha_3, \alpha_2, \alpha_1) \quad (\text{A.47})$$

$$R_{\overleftarrow{\sigma}}(\alpha_1, \alpha_2, \alpha_3) = R_{\overrightarrow{r(\sigma)}}(\alpha_3, \alpha_2, \alpha_1), \quad (\text{A.48})$$

where $r(\sigma) := (\sigma(3) \sigma(2) \sigma(1))$ is the reverse of σ . In words, given the Euler angles of a rotation w. r. t. a moving coordinate frame, the corresponding Euler angles for a fixed coordinate frame can be found by reversing the rotation order and swapping α_1 and α_3 , and vice versa. For example,

$$R_{\overleftarrow{xyz}}(\alpha_1, \alpha_2, \alpha_3) = R_{\overrightarrow{zyx}}(\alpha_3, \alpha_2, \alpha_1). \quad (\text{A.49})$$

It has been mentioned above that the parameter space for the \overleftarrow{xyz} Euler convention is $\mathcal{E}_3 = (-\pi, \pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times (-\pi, \pi]$. This parameter space applies to all Euler conventions that have been studied so far.

In practice, there is a further common way of defining Euler angles. Instead of using three different axes for the successive rotations, it is sufficient to choose two different axes and then alternate rotations about these axes. This leads to the six additional rotation orders $\Sigma_2 := \{xyx, xzx, yxy, yzy, zxz, zyz\}$, which we refer to as *two-axis rotation orders*. Their definition in terms of basic rotations is similar to the three-axis rotation orders. Considering the difference between a fixed and a moving coordinate frame during successive basic rotations, the same considerations as in Sect. A.5.1 apply. For example, in the case of a moving coordinate frame and the xyx rotation order, we obtain

$$\begin{aligned} R_{\overleftarrow{xyx}}(\alpha_1, \alpha_2, \alpha_3) &= R_x(\alpha_1)R_y(\alpha_2)R_x(\alpha_3) \\ &= \begin{pmatrix} c_2 & s_2s_3 & s_2c_3 \\ s_1s_2 & c_1c_3 - s_1c_2s_3 & -c_1s_3 - s_1c_2c_3 \\ -c_1s_2 & s_1c_3 + c_1c_2s_3 & -s_1s_3 + c_1c_2c_3 \end{pmatrix}, \end{aligned} \quad (\text{A.50})$$

where the multiplication order is left-to-right. Conversely, the rotation

$$R_{\overrightarrow{xyx}}(\alpha_1, \alpha_2, \alpha_3) = R_x(\alpha_3)R_y(\alpha_2)R_x(\alpha_1) \quad (\text{A.51})$$

describes the case of a fixed reference frame, corresponding to a right-to-left multiplication order. Since any two-axis rotation order $\sigma \in \Sigma_2$ is symmetric and therefore invariant under string reversal, switching between a moving reference frame and a fixed reference frame amounts to swapping the angles α_1 and α_3 in the corresponding triple of Euler angles.

As a further simplification, any two-axis Euler convention allows for inversion of rotations by directly manipulating the Euler angles. Here is an example for the \overrightarrow{xyx} convention:

$$R_{\overrightarrow{xyx}}^\top(\alpha_1, \alpha_2, \alpha_3) = R_x^\top(\alpha_3)R_y^\top(\alpha_2)R_x^\top(\alpha_1) = R_{\overleftarrow{xyx}}(-\alpha_3, -\alpha_2, -\alpha_1), \quad (\text{A.52})$$

showing that inversion can be achieved by swapping the first and the third Euler angle and then negating all angles. Note that an analogous strategy for inversion would fail for a three-axis rotation order $\sigma \in \Sigma_3$ because transposition leads to a reversal of the basic rotations, which necessitates a change of the Euler convention from σ to $r(\sigma)$.

Similar considerations as in Sect. A.5 show that the parameter space for two-axis Euler conventions σ should be chosen as $\mathcal{E}_2 := (-\pi, \pi] \times [0, \pi] \times (-\pi, \pi]$ due to the symmetries

$$R_{\overrightarrow{\sigma}}(\alpha_1, \alpha_2, \alpha_3) = R_{\overleftarrow{\sigma}}(\pi + \alpha_1, -\alpha_2, \pi + \alpha_3) \quad (\text{A.53})$$

for $\sigma \in \Sigma_2$. The value of the angle α_2 can then be forced into the interval $[0, \pi]$ for any given triple of Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top \in (-\pi, \pi]^3$ by the following method. In case $\alpha_2 \in (-\pi, 0)$, we reparameterize to the angles $(\alpha'_1, \alpha'_2, \alpha'_3)^\top := (\pi + \alpha_1, -\alpha_2, \pi + \alpha_3)^\top$, where angle values beyond the interval $(-\pi, \pi]$ are reduced back into this interval modulo 2π .

A complete list of conversion formulas between the twelve possible left-to-right Euler conventions and other rotational representations will be given in Sect. A.6.1, A.6.3, A.6.2, and A.6.4. Our considerations regarding the relation between right-to-left and left-to-right multiplication orders enable the reader to easily derive the corresponding formulas for the right-to-left Euler conventions.

A.5.3 Geometric Interpretation

We will now show geometrically what has already been shown algebraically: the xyz Euler convention with a fixed reference frame and Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top$ produces the same rotation as the zyx Euler convention with a moving reference frame and Euler angles $(\alpha_3, \alpha_2, \alpha_1)^\top$. Of course, analogous geometrical arguments apply for other pairs of corresponding Euler conventions.

Our considerations are inspired by a mechanical realization of three-dimensional rotations by means of three *gimbals* or *Cardan rings*, see Fig. A.1. A gimbal is basically a planar ring that can rotate about an axis fitted along one of the gimbal's diameters. Now three such gimbals of decreasing sizes are assembled within each other in a specific way:

- the axis of the outer (largest) gimbal is mounted in the direction of the z axis of a fixed reference frame,
- the middle gimbal's axis is y_* ; it is mounted inside the outer gimbal such that y_* is orthogonal to the z axis of the outer gimbal,
- the inner (smallest) gimbal's axis is x_* ; it is mounted inside the middle gimbal such that x_* is orthogonal to the y_* axis of the middle gimbal; the z_* axis is defined as $x_* \times y_*$ and lies within the inner gimbal plane.

In their initial state, shown on the left hand side of Fig. A.1, the movable axes $x_*, y_*, z_* \in \mathbb{R}^3$ coincide with the fixed world coordinate axes x, y, z . Successive rotations move x_*, y_*, z_* while maintaining the mechanically enforced invariants $y_* \perp z, x_* \perp y_*$, and $z_* \perp x_*$. One

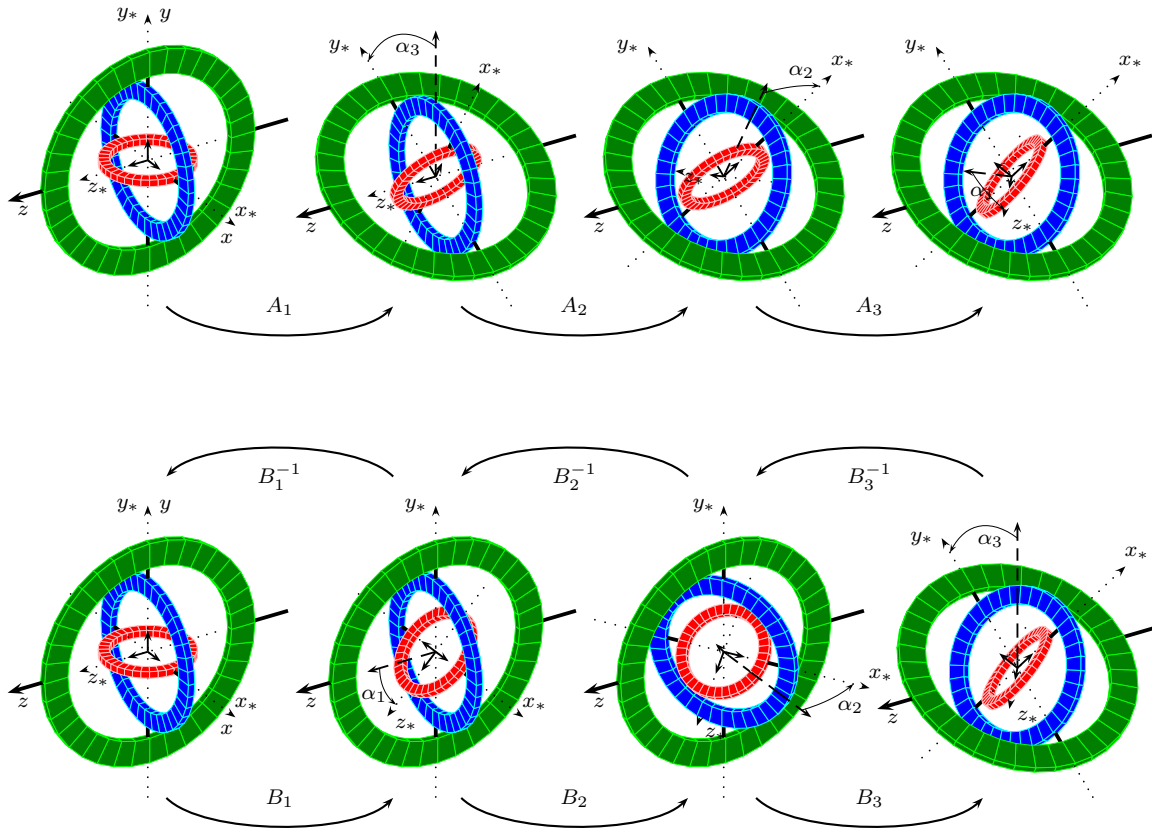


Figure A.1. Geometric interpretation of Euler angles using a three-gimbal assembly. **Sequence A (top row):** Euler convention zyx with a moving reference frame and Euler angles $(\alpha_3, \alpha_2, \alpha_1)^\top = (\frac{\pi}{4}, \frac{\pi}{6}, \frac{\pi}{3})^\top$. **Sequence B (bottom row):** Euler convention xyz with a fixed reference frame and Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top = (\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{4})^\top$. The arrangements on the left hand side represent the initial configuration. Applying successive rotations leads to the arrangements on the right hand side. The axes of each resulting coordinate system are shown within the inner gimbal.

important observation is that a rotation of a larger gimbal also moves the smaller gimbals, since they are rigidly connected. This behavior is an exact simulation of what we have termed “successive rotations with respect to a moving reference frame”. Conversely, a rotation of a smaller gimbal does not affect the larger gimbals. This behavior simulates “successive rotations with respect to a fixed reference frame”. Thus, the zyx Euler convention with a moving reference frame can be simulated by first rotating the outer gimbal, then the middle gimbal, and then the inner gimbal, to which the final coordinate system is attached. The xyz Euler convention with a fixed reference frame can be simulated by first rotating the inner gimbal, then the middle gimbal, and then the outer gimbal. The amount of rotation in each step is given by the corresponding Euler angle.

We now define two sequences of rotations, sequence A and sequence B , which describe the two Euler conventions in terms of the axes x_*, y_*, z_* . Then, the basic idea is to show that after sequence A has been performed, applying the inverse operations of sequence B in reverse order leads back to the initial state, hence sequence A must describe the same rotation as sequence B . The sequences are illustrated by Fig. A.1.

Sequence A: zyx with moving reference frame, Euler angles $(\alpha_3, \alpha_2, \alpha_1)^\top$

step	action	consequences
A_1	rotate about z by α_3	y_* rotates in z^\perp , creating an angle of α_3 between y_* and y , x_* and z_* follow the rotation, the absolute position of y_* does not change henceforth.
A_2	rotate about y_* by α_2	x_* rotates in y_*^\perp by an angle of α_2 , z_* follows the rotation, the absolute position of x_* does not change henceforth.
A_3	rotate about x_* by α_1	z_* rotates in x_*^\perp by an angle of α_1 , $(0; x_*, z_*, z_* \times x_*)$ is the final coordinate system.

Sequence B: xyz with fixed reference frame, Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top$

step	action	consequences
B_1	rotate about x_* by α_1	z_* rotates in x_*^\perp by an angle of α_1 .
B_2	rotate about y_* by α_2	x_* rotates in y_*^\perp by an angle of α_2 , z_* follows the rotation.
B_3	rotate about z by α_3	y_* rotates in z^\perp , creating an angle of α_3 between y_* and y , x_* and z_* follow the rotation, $(0; x_*, z_*, z_* \times x_*)$ is the final coordinate system.

Now we apply the inverse operations B_3^{-1} , B_2^{-1} , B_1^{-1} , in this sequence. After step A_3 , we know that the x_* axis encloses an angle of $\frac{\pi}{2} - \alpha_2$ with the z axis, and the z_* axis encloses an angle of $\frac{\pi}{2} - \alpha_1$ with the y_* axis. In addition, y_* is still in the same fixed position that was established in step A_1 : y_* and y enclose an angle of α_3 in the z^\perp plane. The same position for y_* is also established by step B_3 of sequence B , so performing B_3^{-1} brings the outer gimbal into the state that would be observed if B_1 , B_2 had been performed: y_* and y are aligned. Furthermore, B_3^{-1} moves the middle and the inner gimbal along with the outer gimbal, thus maintaining the relative angles between the gimbals that have been established in steps A_2 and A_3 . Therefore, performing B_2^{-1} undoes the rotation of α_2 about y_* and realigns x_* with x , while leaving the relative angle of the inner gimbal unchanged and also leaving y_* and y aligned. Finally, we can perform B_1^{-1} , which undoes the rotation of α_1 about x_* and realigns z_* with z . In summary, we have established the initial state where x_* , y_* , z_* coincide with x , y , z , respectively. Therefore, sequence A describes the same rotation as sequence B . \square

Gimbal Assembly for Two-Axis Euler Conventions. A simple modification of the gimbal assembly described above allows us to simulate two-axis rotation orders such as xyx or zyz . For xyx with a fixed reference frame, we simply change the neutral position shown in Fig. A.1 in the following way: we pivot the outer gimbal by $\frac{\pi}{2}$ about the y axis while keeping the middle and inner gimbals fixed, see Fig. A.2 (a). This lines up the x_* axis with the outer gimbal's axis. Successive rotations starting at the inner gimbal and progressing outwards then simulate the \overrightarrow{xyx} Euler convention. Similarly, a gimbal assembly for the \overrightarrow{zyz} Euler convention is obtained from the neutral position shown in Fig. A.1 by pivoting the middle gimbal by $\frac{\pi}{2}$ about the y axis, so that the z_* axis and the outer gimbal's axis line up.

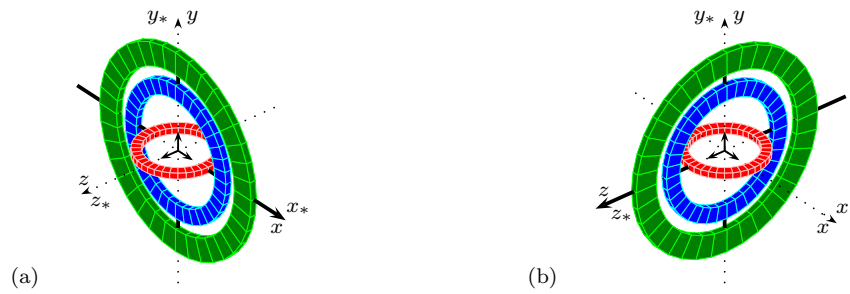


Figure A.2. Gimbal assembly for (a) the \overline{xyx} Euler convention, and for (b) the \overline{zyz} Euler convention. All Euler angles are zero.

A.5.4 Gimbal Lock

Gimbal lock is one of the major drawbacks associated with Euler angles. For three-axis Euler conventions such as \overline{xyz} , gimbal lock occurs whenever the angle α_2 assumes one of the values $\pm\frac{\pi}{2}$, see Fig. A.3. In such a *gimbal lock configuration*, the outer and the middle gimbal are clamped together in a common plane. For $\alpha_2 = +\frac{\pi}{2}$, the x_* axis points in the direction of the $-z$ axis, so a rotation about the x_* axis by α has the same effect as a rotation about the z axis by $-\alpha$. Similarly, for $\alpha_2 = -\frac{\pi}{2}$, the x_* axis points in the direction of the z axis, so a rotation about the x_* axis by α has the same effect as a rotation about the z axis by α . These situations lead to the following problems:

- There are infinitely many gimbal configurations describing the same rotation.
- One rotational DOF is lost, since two free parameters describe a one DOF rotation about the fixed z axis. Specifically, the lost rotational DOF is the ability to rotate about the normal of the outer/middle gimbal plane, since all available axes of rotation are coplanar within that plane.
- As a consequence, a time-varying rotation of constant angular speed that runs through a gimbal lock configuration may require infinitely rapid adjustments of the Euler angles.

For example, starting from the gimbal lock configuration in Fig. A.3 (a), it is mechanically impossible to rotate about the x axis (currently aligned with the z_* axis), unless one first moves the outer and the middle gimbal to the position shown in Fig. A.3 (c), which can be done without changing the resulting rotation (i. e., the inner gimbal can be held fixed while rotating the middle and the outer gimbal). This effect is very undesirable in mechanical systems and gave rise to the term *gimbal lock*. Early gyroscopic devices for attitude tracking during spaceflight, as used in the Apollo space missions [All00], were based on our three-gimbal construction. If the spacecraft moved into a certain “forbidden” orientation in space, gimbal lock would occur, and subsequent rotations would not be trackable. Later, a fourth, actively controlled gimbal was added, which guaranteed that gimbal lock could be avoided in all possible orientations.

A popular description of gimbal lock that can be found in the literature, see, for example, [Gra98, Sho85], simply states that “the x and the z axis become aligned, therefore one rotational DOF is lost.” At first sight, this imprecise statement seems like a contradiction if one thinks of xyz rotations with respect to a fixed reference frame. How can the x and the z axis be aligned if they remain fixed? Here, the solution is that the authors are actually

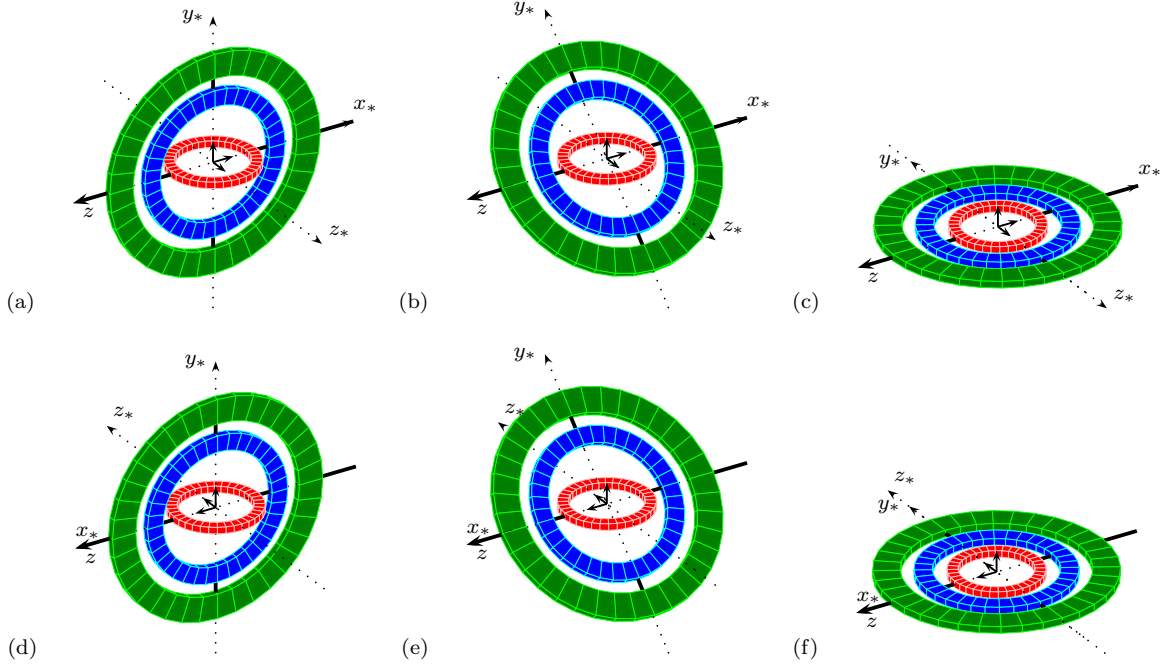


Figure A.3. Different gimbal lock configurations for the \overleftarrow{xyz} Euler convention. (a)–(c) describe the same rotation and (d)–(f) describe the same rotation, respectively. (a) $(0, \frac{\pi}{2}, 0)^\top$, (b) $(\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4})^\top$, (c) $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2})^\top$, (d) $(0, -\frac{\pi}{2}, 0)^\top$, (e) $(-\frac{\pi}{4}, -\frac{\pi}{2}, \frac{\pi}{4})^\top$, (f) $(\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2})^\top$.

referring to the x_* axis and the z axis as becoming aligned. Regardless of whether successive basic rotations are expressed in a fixed or in a moving reference frame, one can imagine a reference frame that rotates along with the rotating object. This reference frame is shown in Figs. A.2, A.1, and A.3 as the coordinate system $(0; x_*, z_*, z_* \times x_*)$ drawn within the inner gimbal. Here, the x_* axis always represents the image of the x axis under the basic rotations that have been applied so far and therefore serves as a memory aid during the third basic rotation about z : the x_* axis tells us from the rotating object's point of view about which axis it has already been rotated. Now if the x_* axis becomes aligned with the z axis, the object is rotated twice about the same axis. Note that the y_* axis cannot be become aligned with the z axis since it always satisfies the orthogonality constraint $y_* \perp z$.

We will now investigate gimbal lock algebraically for $\alpha_2 = +\frac{\pi}{2}$. Eqn. (A.33) gives us

$$R_{\overleftarrow{xyz}}\left(\alpha_1, \frac{\pi}{2}, \alpha_3\right) = \begin{pmatrix} 0 & -\sin \alpha_3 \cos \alpha_1 + \cos \alpha_3 \sin \alpha_1 & \sin \alpha_3 \sin \alpha_1 + \cos \alpha_3 \cos \alpha_1 \\ 0 & \cos \alpha_3 \cos \alpha_1 + \sin \alpha_3 \sin \alpha_1 & -\cos \alpha_3 \sin \alpha_1 + \sin \alpha_3 \cos \alpha_1 \\ -1 & 0 & 0 \end{pmatrix}. \quad (\text{A.54})$$

Applying the trigonometric identities

$$\cos(\beta - \gamma) = \cos \beta \cos \gamma + \sin \beta \sin \gamma \quad (\text{A.55})$$

$$\sin(\beta - \gamma) = \sin \beta \cos \gamma - \cos \beta \sin \gamma, \quad (\text{A.56})$$

we obtain

$$R_{\overline{xyz}}\left(\alpha_1, \frac{\pi}{2}, \alpha_3\right) = \begin{pmatrix} 0 & \sin(\alpha_1 - \alpha_3) & \cos(\alpha_1 - \alpha_3) \\ 0 & \cos(\alpha_1 - \alpha_3) & -\sin(\alpha_1 - \alpha_3) \\ -1 & 0 & 0 \end{pmatrix}. \quad (\text{A.57})$$

The following factorizations provide an interpretation of this matrix:

$$\begin{aligned} R_{\overline{xyz}}\left(\alpha_1, \frac{\pi}{2}, \alpha_3\right) &= \begin{pmatrix} \cos(\alpha_1 + \alpha_3) & \sin(\alpha_1 - \alpha_3) & 0 \\ -\sin(\alpha_1 - \alpha_3) & \cos(\alpha_1 - \alpha_3) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \\ &= R_z^\top(\alpha_1 - \alpha_3) R_y\left(\frac{\pi}{2}\right) = R_z(\alpha_3 - \alpha_1) R_y\left(\frac{\pi}{2}\right), \end{aligned} \quad (\text{A.58})$$

$$\begin{aligned} \text{or } R_{\overline{xyz}}\left(\alpha_1, \frac{\pi}{2}, \alpha_3\right) &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_1 - \alpha_3) & -\sin(\alpha_1 - \alpha_3) \\ 0 & \sin(\alpha_1 - \alpha_3) & \cos(\alpha_1 - \alpha_3) \end{pmatrix} \\ &= R_y\left(\frac{\pi}{2}\right) R_x(\alpha_1 - \alpha_3). \end{aligned} \quad (\text{A.59})$$

Hence, the matrix $R_{\overline{xyz}}\left(\alpha_1, \frac{\pi}{2}, \alpha_3\right)$ describes a rotation about the x axis by an angle of $\alpha_1 - \alpha_3$ followed by a rotation about the y axis by an angle of $\frac{\pi}{2}$. Alternatively, the matrix can be interpreted as a rotation about the y axis by an angle of $\frac{\pi}{2}$ followed by a rotation about the z axis by an angle of $\alpha_3 - \alpha_1$.

As a result, we note that all Euler angles in the one-dimensional set

$$M_{\overline{xyz}}^+(\beta) := \left\{ \left(\alpha_1, +\frac{\pi}{2}, \alpha_3 \right)^\top \mid \alpha_1 - \alpha_3 = \beta \right\} \subset \mathcal{E}_3 \quad (\text{A.60})$$

describe the same rotation. Similar interpretations can be derived for the case $\alpha_2 = -\frac{\pi}{2}$, yielding the one-dimensional set

$$M_{\overline{xyz}}^-(\beta) := \left\{ \left(\alpha_1, -\frac{\pi}{2}, \alpha_3 \right)^\top \mid \alpha_1 + \alpha_3 = \beta \right\} \subset \mathcal{E}_3 \quad (\text{A.61})$$

of Euler angles leading to the same rotation. Since we want our Euler parametrization to be bijective, we remove the sets $M_{\overline{xyz}}^+(\beta)$ and $M_{\overline{xyz}}^-(\beta)$ from the parameter space \mathcal{E}_3 for all $\beta \in (-\pi, \pi]$ up to a single representative $(0, \pm\frac{\pi}{2}, \beta)^\top$ from each set:

$$\tilde{\mathcal{E}}_3 := \mathcal{E}_3 \setminus \bigcup_{\beta \in (-\pi, \pi]} \left(M_{\overline{xyz}}^\pm(\beta) \setminus \left\{ (0, \pm\frac{\pi}{2}, \beta)^\top \right\} \right). \quad (\text{A.62})$$

Other three-axis Euler conventions σ have different properties regarding the sets $M_\sigma^+(\beta)$ and $M_\sigma^-(\beta)$. For example, the defining relations $\alpha_1 - \alpha_3 = \beta$ and $\alpha_1 + \alpha_3 = \beta$ must be interchanged for the \overline{zxy} convention, which we will encounter in a real-world example later in this section. For two-axis rotation orders such as xyx or zyz , gimbal lock occurs for $\alpha_2 \in \{0, \pi\}$. This becomes immediately clear by comparing Fig. A.2 and Fig. A.3 (a), which represent the same gimbal configuration.

The gimbal lock phenomenon results from a more fundamental problem attached to any parametrization of SO_3 that uses three parameters. The function $R_{\overline{xyz}} : \mathbb{R}^3 \rightarrow \text{SO}_3$ as defined in (A.33) is continuous and surjective. To obtain a unique Euler representation for a rotation,

we had to restrict the parameter space to a subset $\tilde{\mathcal{E}}_3 \subset \mathbb{R}^3$. Here, it can be shown that there is no choice for $\tilde{\mathcal{E}}_3$ such that $R_{\overleftarrow{xyz}}$ restricted to $\tilde{\mathcal{E}}_3$ is bijective with a continuous inverse $R_{\overleftarrow{xyz}}^{-1} : \text{SO}_3 \rightarrow \tilde{\mathcal{E}}_3$. This is due to a well-known topological fact stating that SO_3 is not homeomorphic to any subset of \mathbb{R}^3 .

Experiment. To demonstrate the effects associated with gimbal lock, we consider the three unit quaternions $q_1 = (1, 0, 0, 0)^\top$, $q_2 = (\frac{1}{2}\sqrt{2}, 0, \frac{1}{2}\sqrt{2}, 0)^\top$, and $q_3 = (\frac{1}{2}\sqrt{2}, 0, 0, \frac{1}{2}\sqrt{2})^\top$. The first and the third quaternion correspond to the \overleftarrow{xyz} Euler angles $p_1 = (0, 0, 0)^\top$ and $p_3 = (0, 0, \frac{\pi}{2})^\top$, respectively. The second quaternion corresponds to the set of gimbal lock angles $M_{\overleftarrow{xyz}}^+(0) = \left\{ (\alpha_1, \frac{\pi}{2}, \alpha_3)^\top \mid \alpha_1 = \alpha_3 \right\}$, which we had removed from the parameter space \mathcal{E}_3 up to the representative $(0, \frac{\pi}{2}, 0)^\top$.

Now we use spherical linear interpolation as defined in Sect. A.3 to construct a continuous path $q : [0, 1] \rightarrow S^3$ with $q(0) = q(1) = q_1$, $q(\frac{1}{3}) = q_2$, and $q(\frac{2}{3}) = q_3$. The path q connects q_1 with q_2 , q_2 with q_3 , and q_3 with q_1 into a closed round-trip path, which describes a continuous change of orientation with constant angular speed along each segment. Fig. A.4 shows a projection of the resulting path into \mathbb{R}^3 . Note that we only compute a regularly sampled version of the path comprising 3,000 samples. Next, we apply the inverse of the \overleftarrow{xyz} Euler parametrization to obtain the path $p := R_{\overleftarrow{xyz}}^{-1} \circ q : [0, 1] \rightarrow \mathcal{E}_3$. It turns out that the image of p in the Euler angle space, labeled in Fig. A.5 as $\Delta\alpha_2^{(4)} = 0^\circ$, is disconnected: on the way from p_2 to p_3 the path first runs to $p_4 = (\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4})^\top$. The path would now lead to p_3 along the line $M_{\overleftarrow{xyz}}^+(0)$ in a continuous fashion, but with infinite velocity. However, since we removed $M_{\overleftarrow{xyz}}^+(0)$ from the parameter space, the path has a discontinuity at $t = \frac{1}{3}$, so there is a jump to $p_2^{(4)} = (0, \frac{\pi}{2}, 0)^\top$. This reflects the discontinuity of $R_{\overleftarrow{xyz}}^{-1}$ near gimbal lock configurations.

To investigate the behavior of $R_{\overleftarrow{xyz}}^{-1}$ in the neighborhood of this discontinuity, we construct a sequence of paths $q^{(i)}$ on S^3 leading through q_1 , $q_2^{(i)}$, and q_3 , where $q_2^{(i)}$ approaches $q_2 = (\frac{1}{2}\sqrt{2}, 0, \frac{1}{2}\sqrt{2}, 0)^\top$ from a certain direction for increasing i . We choose this direction in such a way that $p_2^{(i)} := R_{\overleftarrow{xyz}}^{-1}(q_2^{(i)}) = (0, \frac{\pi}{2} - \Delta\alpha_2^{(i)}, 0)^\top$ for $\Delta\alpha_2^{(i)} > 0$. Three resulting paths $p^{(i)}$ are shown in Fig. A.5 (a)–(c) for different values of $\Delta\alpha_2^{(i)}$. The closer the point $p_2^{(i)}$ approaches $p_2^{(4)}$, the larger the velocity gets along the connection between $p_2^{(i)}$ and p_3 . Here, the fact that we work with regularly sampled versions of the paths $q^{(i)}$ enables us to estimate the velocity along the path by means of the distance between subsequent dots describing the path.

The discontinuity of $R_{\overleftarrow{xyz}}^{-1}$ near gimbal lock configurations is similar to the discontinuity of the inverse of the Earth's latitude/longitude parametrization at the north pole and the south pole: with a latitude near $\pm 90^\circ$, moving a small distance in east or west direction requires very large changes in longitude. For a latitude of exactly $\pm 90^\circ$, the longitude can vary without affecting the current position on the Earth's surface.

Avoiding Gimbal Lock by Reparametrization. In 3D graphics and animation systems such as MAYA, Euler angles are used as a GUI tool for interactively specifying 3D rotations. Here, the gimbal lock effect would lead to very unintuitive behavior: the user could change a parameter value without any effect on the rotation. Therefore, the Euler convention is switched as soon as α_2 gets closer to 90° than some threshold, say, 5° . For example, if the

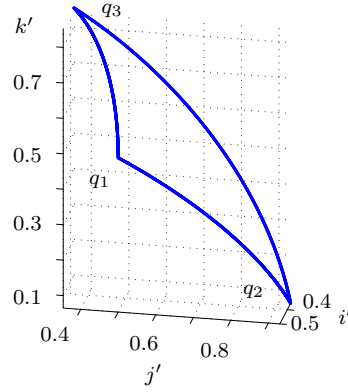


Figure A.4. A closed path on S^3 connecting three unit quaternions q_1, q_2, q_3 by means of spherical linear interpolation. The 3D visualization has been obtained by orthogonally projecting \mathbb{R}^4 onto the subspace $(\text{span}\{(1, 1, 1, 1)^\top\})^\perp$, the axes of which are denoted by i', j' , and k' .

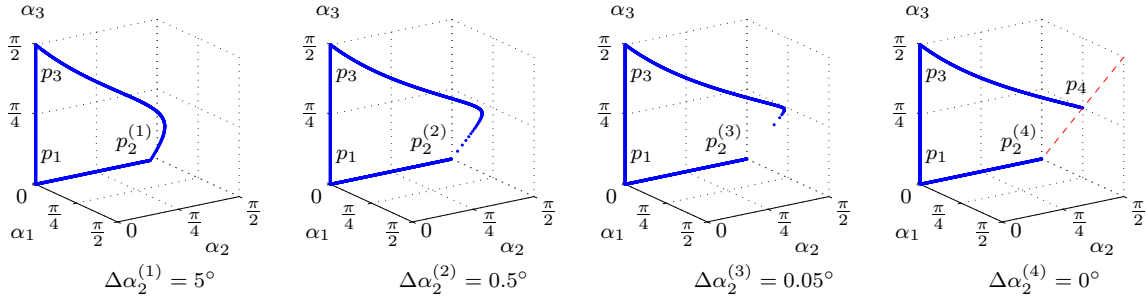


Figure A.5. Minor variations of the closed path shown in Fig. A.4 are transformed to Euler angles by means of the inverse of the \overleftarrow{xyz} Euler parametrization, yielding paths in the Euler angle space, \mathcal{E}_3 . From left to right, the points $p_2^{(i)} := (0, \frac{\pi}{2} - \Delta\alpha_2^{(i)}, 0)^\top$, $i = 1, \dots, 4$, successively approach the gimbal lock configuration $(0, \frac{\pi}{2}, 0)^\top$. By $\Delta\alpha_2^{(i)}$, we denote the distance of Euler angle α_2 to the critical configuration. The discontinuity of the parametrization's inverse leads to a disrupted path for $\Delta\alpha_2^{(4)} = 0^\circ$. The red dashed line indicates a part of $M_{\overleftarrow{xyz}}^+(0)$.

initial Euler convention is \overleftarrow{xyz} and the current Euler angles are $(0^\circ, 86^\circ, 0^\circ)^\top$, one could reparametrize to the \overleftarrow{xzy} convention and $(0^\circ, 0^\circ, 86^\circ)^\top$. Note that in general, the conversion between Euler conventions is not as straightforward as in this example. The simplest way is to convert to quaternions (see Sect. A.6.2) and then back to Euler angles (see Sect. A.6.4).

Real-world Examples. Fig. A.6 shows \overrightarrow{zxy} Euler angle and quaternion trajectories describing the right knee angle for a 30-second modern dance motion. The Euler angles have been taken directly from a BVH motion capture file (cf. Sect. D.2), whereas the quaternions have been derived by the conversion formulas given in Sect. A.6.2. Since the knee joint has only one DOF, since the axis of rotation for the knee coincides with the x axis, and since the makers of this particular motion capture file decided to use the \overrightarrow{zxy} convention, this is a good setting to observe gimbal locks. We are dealing with a one DOF joint, so there should be constant rotation or no rotation at all about the z and the y axis. Indeed, the angles α_1 and α_3 are close close to zero most of the time. However, as the angle α_2 gets close to the value $\frac{\pi}{2}$ for certain frames, the angles α_1 and α_3 assume none-zero values. This effect is particularly strong around frames 800 and 1,450. Here, both α_1 and α_3 assume large absolute values of

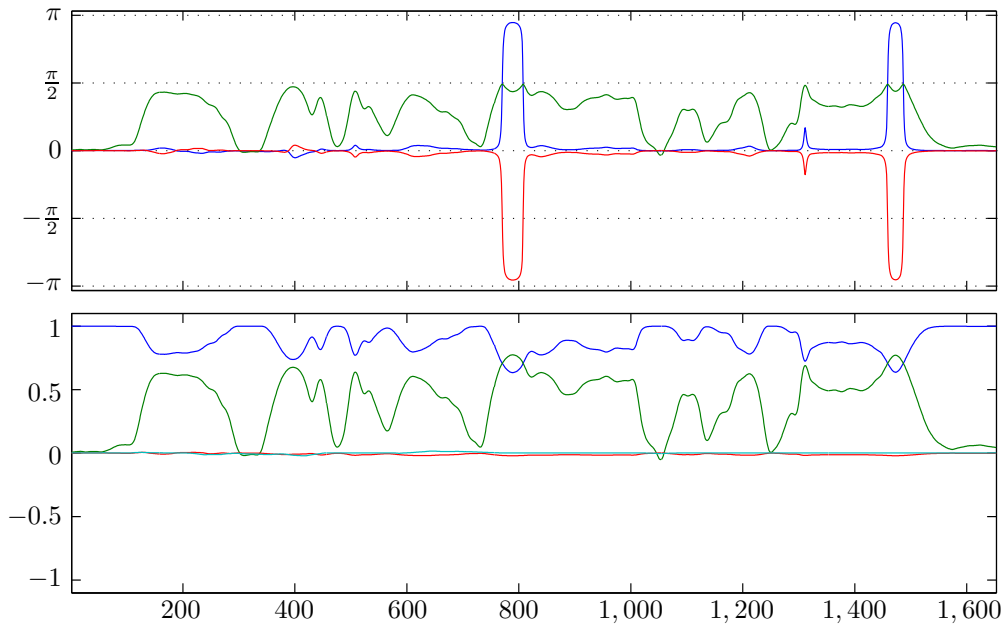


Figure A.6. \overrightarrow{zxy} Euler angles (top) and quaternions (bottom) for the right knee angle in a 1,653-frame modern dance motion, sampled at 60 Hz. The parameters are color-coded. Euler angles: blue (α_1), green (α_2), and red (α_3). Quaternions: blue (w), green (x), red (y), and cyan (z). The horizontal axis represents time in frames.

opposing signs, which effectively cancel out since the z axis is aligned or nearly aligned with the y axis in this situation. A perfect one DOF joint rotating about the x axis would keep the Euler angles for z and y at zero. However, the Euler angles in Fig. A.6 have been obtained from an inverse kinematics solver, which seems to have had no constraints on the Euler angles for gimbal lock configurations. Comparing the Euler angles to the corresponding quaternions, no such discontinuous behavior can be observed.

Now imagine comparing this mocap data stream to a second, very similar instance of this dance motion by means of their Euler angles. It is very probable that the large values of α_1 and α_3 around certain frames would not occur in this way for the second instance, even if the corresponding knee angles were very close to each other. This would strongly support the incorrect decision that the two motions are dissimilar. Therefore, Euler angles are not suitable for the comparison of motion capture data.

Discussion. In summary, Euler angles are a very compact and intuitive representation of rotations, which have a direct geometric interpretation and facilitate a semantically meaningful way to constrain rotations by simply constraining the domain of the Euler angles. On the other hand, due to the discontinuities in the inverse of the parametrization, computing with Euler angles can lead to unforeseen artifacts. In particular, comparing rotations via their Euler angles may be problematic—near or at gimbal lock configurations, small variations of the rotation are reflected in large variations of the Euler angles. Also, interpolation of rotations via Euler angles typically leads to rotations with non-constant angular velocity, which is perceptually disturbing. In view of such tasks, some of the representations that have been discussed in the previous sections are more suitable.

Nevertheless, Euler angles are frequently used as an input tool in animation and graphics applications such as manual keyframe animation, postprocessing of motion capture data, or construction of 3D models. Here, they provide mostly intuitive and interactive three-dimensional control over rotations, since each of the angles has an easy interpretation and can be visualized by appropriate methods. We have seen that Euler angles are typically used to encode mocap data using skeleton-based mocap file formats such as BVH or ASF/AMC, see also App. D.

Certain Euler conventions allow for easy inversion of rotations in terms of Euler angles, but composing rotations in terms of Euler angles is generally not possible. Finally, Euler angles are suited for solving differential equations involving rotations, and for differentiation and optimization purposes, but only as long as the specific application only involves rotations that do not correspond to gimbal lock configurations, see [Gra98].

A.6 Compendium of Conversion Formulas

This section provides a compendium of explicit conversion formulas for converting Euler angles to and from rotation matrices and unit quaternions. As an extension to the collections in [Sho85, Sho94], explicit conversions for all Euler conventions are given. These formulas are essential for parsing, writing, and editing motion capture files in kinematic-chain-based formats such as ASF/AMC or BVH, see App. D.

A.6.1 Euler Angles to Matrix

The Euler-to-matrix conversion formulas for the left-to-right multiplication order can be found in Tab. A.2. We focus on the left-to-right multiplication order (corresponding to a moving reference frame) since this is the type of convention used in the popular BVH mocap file format. As we know from (A.47), we have $R_{xyz}(\alpha_1, \alpha_2, \alpha_3) = R_{zyx}(\alpha_3, \alpha_2, \alpha_1)$, so the corresponding formulas for the right-to-left multiplication order can be read off by reversing the rotation order and swapping α_1 with α_3 . As usual, we abbreviate the trigonometric functions in the following way:

$$c_i := \cos \alpha_i \quad \text{and} \quad s_i := \sin \alpha_i \quad \text{for } i = 1, 2, 3. \quad (\text{A.63})$$

A.6.2 Euler Angles to Quaternion

As we know from Sect. A.3, the scalar part s of a unit quaternion $(s, v) \in \mathbb{H}$ is the cosine of half the angle of rotation, and the vector part v points along the axis of rotation, where the length of v is the sine of half the angle of rotation. Therefore, the three basic Euler rotations (A.30)–(A.31) have the following quaternion representations:

$$q_x(\alpha_1) = \left(\cos \frac{\alpha_1}{2}, \sin \frac{\alpha_1}{2}, 0, 0 \right)^\top \quad (\text{A.76})$$

$$q_y(\alpha_2) = \left(\cos \frac{\alpha_2}{2}, 0, \sin \frac{\alpha_2}{2}, 0 \right)^\top \quad (\text{A.77})$$

$$q_z(\alpha_3) = \left(\cos \frac{\alpha_3}{2}, 0, 0, \sin \frac{\alpha_3}{2} \right)^\top \quad (\text{A.78})$$

$$R_{\overrightarrow{xyz}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_2 c_3 & -c_2 s_3 & s_2 \\ s_1 s_2 c_3 + c_1 s_3 & -s_1 s_2 s_3 + c_1 c_3 & -s_1 c_2 \\ -c_1 s_2 c_3 + s_1 s_3 & c_1 s_2 s_3 + s_1 c_3 & c_1 c_2 \end{pmatrix} \quad (\text{A.64})$$

$$R_{\overrightarrow{xzy}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_2 c_3 & -s_2 & c_2 s_3 \\ s_1 s_3 + c_1 s_2 c_3 & c_1 c_2 & -s_1 c_3 + c_1 s_2 s_3 \\ -c_1 s_3 + s_1 s_2 c_3 & s_1 c_2 & c_1 c_3 + s_1 s_2 s_3 \end{pmatrix} \quad (\text{A.65})$$

$$R_{\overrightarrow{yxz}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_1 c_3 + s_1 s_2 s_3 & -c_1 s_3 + s_1 s_2 c_3 & s_1 c_2 \\ c_2 s_3 & c_2 c_3 & -s_2 \\ -s_1 c_3 + c_1 s_2 s_3 & s_1 s_3 + c_1 s_2 c_3 & c_1 c_2 \end{pmatrix} \quad (\text{A.66})$$

$$R_{\overrightarrow{yzx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_1 c_2 & -c_1 s_2 c_3 + s_1 s_3 & c_1 s_2 s_3 + s_1 c_3 \\ s_2 & c_2 c_3 & -c_2 s_3 \\ -s_1 c_2 & s_1 s_2 c_3 + c_1 s_3 & -s_1 s_2 s_3 + c_1 c_3 \end{pmatrix} \quad (\text{A.67})$$

$$R_{\overrightarrow{zxy}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} -s_1 s_2 s_3 + c_1 c_3 & -s_1 c_2 & s_1 s_2 c_3 + c_1 s_3 \\ c_1 s_2 s_3 + s_1 c_3 & c_1 c_2 & -c_1 s_2 c_3 + s_1 s_3 \\ -c_2 s_3 & s_2 & c_2 c_3 \end{pmatrix} \quad (\text{A.68})$$

$$R_{\overrightarrow{zyx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_1 c_2 & -s_1 c_3 + c_1 s_2 s_3 & s_1 s_3 + c_1 s_2 c_3 \\ s_1 c_2 & c_1 c_3 + s_1 s_2 s_3 & -c_1 s_3 + s_1 s_2 c_3 \\ -s_2 & c_2 s_3 & c_2 c_3 \end{pmatrix} \quad (\text{A.69})$$

$$R_{\overrightarrow{xyx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_2 & s_2 s_3 & s_2 c_3 \\ s_1 s_2 & c_1 c_3 - s_1 c_2 s_3 & -c_1 s_3 - s_1 c_2 c_3 \\ -c_1 s_2 & s_1 c_3 + c_1 c_2 s_3 & -s_1 s_3 + c_1 c_2 c_3 \end{pmatrix} \quad (\text{A.70})$$

$$R_{\overrightarrow{xzx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_2 & -s_2 c_3 & s_2 s_3 \\ c_1 s_2 & -s_1 s_3 + c_1 c_2 c_3 & -c_1 c_2 s_3 - s_1 c_3 \\ s_1 s_2 & s_1 c_2 c_3 + c_1 s_3 & c_1 c_3 - s_1 c_2 s_3 \end{pmatrix} \quad (\text{A.71})$$

$$R_{\overrightarrow{yxy}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_1 c_3 - s_1 c_2 s_3 & s_1 s_2 & s_1 c_2 c_3 + c_1 s_3 \\ s_2 s_3 & c_2 & -s_2 c_3 \\ -c_1 c_2 s_3 - s_1 c_3 & c_1 s_2 & -s_1 s_3 + c_1 c_2 c_3 \end{pmatrix} \quad (\text{A.72})$$

$$R_{\overrightarrow{yzx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} -s_1 s_3 + c_1 c_2 c_3 & -c_1 s_2 & c_1 c_2 s_3 + s_1 c_3 \\ s_2 c_3 & c_2 & s_2 s_3 \\ -s_1 c_2 c_3 - c_1 s_3 & s_1 s_2 & c_1 c_3 - s_1 c_2 s_3 \end{pmatrix} \quad (\text{A.73})$$

$$R_{\overrightarrow{zxx}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_1 c_3 - s_1 c_2 s_3 & -s_1 c_2 c_3 - c_1 s_3 & s_1 s_2 \\ c_1 c_2 s_3 + s_1 c_3 & -s_1 s_3 + c_1 c_2 c_3 & -c_1 s_2 \\ s_2 s_3 & s_2 c_3 & c_2 \end{pmatrix} \quad (\text{A.74})$$

$$R_{\overrightarrow{zyz}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} -s_1 s_3 + c_1 c_2 c_3 & -c_1 c_2 s_3 - s_1 c_3 & c_1 s_2 \\ s_1 c_2 c_3 + c_1 s_3 & c_1 c_3 - s_1 c_2 s_3 & s_1 s_2 \\ -s_2 c_3 & s_2 s_3 & c_2 \end{pmatrix} \quad (\text{A.75})$$

Table A.2. Euler-to-matrix conversion for the twelve Euler conventions with left-to-right multiplication order.

Input: Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top$ and an Euler convention.

Output: corresponding rotation matrix $R \in \text{SO}_3$.

for $\alpha \in (-\pi, \pi]$. Multiplying these basic quaternions together in the respective order yields the twelve Euler angle to quaternion conversion formulas given in Tab. A.3. For example, the conversion for the \overline{xyz} Euler convention is as follows:

$$\begin{aligned} q_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3) &= q_x(\alpha_1)q_y(\alpha_2)q_z(\alpha_3) \\ &= \begin{pmatrix} \cos(\frac{\alpha_1}{2}) \cos(\frac{\alpha_2}{2}) \cos(\frac{\alpha_3}{2}) - \sin(\frac{\alpha_1}{2}) \sin(\frac{\alpha_2}{2}) \sin(\frac{\alpha_3}{2}) \\ \sin(\frac{\alpha_1}{2}) \cos(\frac{\alpha_2}{2}) \cos(\frac{\alpha_3}{2}) + \cos(\frac{\alpha_1}{2}) \sin(\frac{\alpha_2}{2}) \sin(\frac{\alpha_3}{2}) \\ \cos(\frac{\alpha_1}{2}) \sin(\frac{\alpha_2}{2}) \cos(\frac{\alpha_3}{2}) - \sin(\frac{\alpha_1}{2}) \cos(\frac{\alpha_2}{2}) \sin(\frac{\alpha_3}{2}) \\ \cos(\frac{\alpha_1}{2}) \cos(\frac{\alpha_2}{2}) \sin(\frac{\alpha_3}{2}) + \sin(\frac{\alpha_1}{2}) \sin(\frac{\alpha_2}{2}) \cos(\frac{\alpha_3}{2}) \end{pmatrix} \end{aligned} \quad (\text{A.79})$$

A.6.3 Matrix to Euler Angles

To demonstrate the derivation scheme for the following twelve matrix-to-Euler conversion formulas, we consider the \overline{xyz} convention as an example. Given an arbitrary rotation matrix $R = (r_{ij}) \in \text{SO}_3$, we perform comparison of coefficients with the \overline{xyz} Euler-to-matrix conversion, (A.64):

$$R_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3) = \begin{pmatrix} c_2c_3 & -c_2s_3 & s_2 \\ s_1s_2c_3 + c_1s_3 & -s_1s_2s_3 + c_1c_3 & -s_1c_2 \\ -c_1s_2c_3 + s_1s_3 & c_1s_2s_3 + s_1c_3 & c_1c_2 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \quad (\text{A.92})$$

which gives us $s_2 = r_{13}$, or

$$\alpha_2 = \arcsin r_{13} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]. \quad (\text{A.93})$$

As long as $\alpha_2 \neq \pm\frac{\pi}{2}$, computing the Euler angles is straightforward. Using the identities $s_1c_2 = -r_{23}$, $c_1c_2 = r_{33}$, $s_3c_2 = -r_{12}$, and $c_3c_2 = r_{11}$, we can apply the two-argument inverse tangent, see Fig. A.7,

$$\arctan2 : \mathbb{R}^2 \setminus \{0\} \rightarrow (-\pi, \pi], \quad \arctan2(y, x) := \begin{cases} \arctan \frac{y}{x} & \text{if } x > 0 \\ \arctan \frac{y}{x} + \pi & \text{if } x < 0 \\ \frac{\pi}{2} & \text{if } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0, y < 0 \\ \text{undefined} & \text{otherwise,} \end{cases} \quad (\text{A.94})$$

where $\arctan : \mathbb{R} \rightarrow (-\frac{\pi}{2}, \frac{\pi}{2})$, to obtain

$$\alpha_1 = \arctan2(s_1c_2, c_1c_2) = \arctan2(-r_{23}, r_{33}), \quad (\text{A.95})$$

$$\alpha_3 = \arctan2(s_3c_2, c_3c_2) = \arctan2(-r_{12}, r_{11}), \quad (\text{A.96})$$

since the terms $c_2 \neq 0$ cancel out in the arguments of the inverse tangent functions.

The case $\alpha_2 = \pm\frac{\pi}{2} \Rightarrow c_2 = 0$ corresponds to a gimbal lock configuration where the effect of the Euler angles α_1 and α_3 cannot be distinguished, see Sect. A.5.4. We resolve this situation by the arbitrary (but convenient) choice of $\alpha_1 = 0$, or $c_1 = 1$, $s_1 = 0$. This greatly simplifies the matrix entries at positions (2, 1) and (2, 2) in $R_{\overline{xyz}}(\alpha_1, \alpha_2, \alpha_3)$, yielding $s_3 = r_{21}$ and $c_3 = r_{22}$. Thus, we obtain for the case $\alpha_2 = \pm\frac{\pi}{2}$

$$\alpha_1 = 0, \quad (\text{A.97})$$

$$\alpha_3 = \arctan2(s_3, c_3) = \arctan2(r_{21}, r_{22}). \quad (\text{A.98})$$

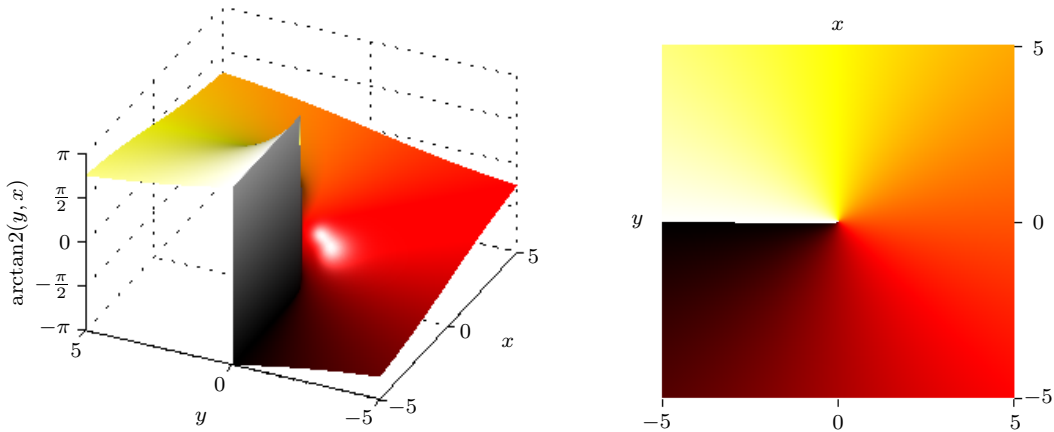


Figure A.7. Plots of the two-argument arctan2 function. The function returns the angle enclosed between the argument vector and the x axis, restricted to the range $(-\pi, \pi]$. **Left:** 3D plot of arctan2 . **Right:** Color-coded 2D plot of arctan2 . The colors range from black $(-\pi)$ over red (0) to white (π) .

For the Euler conventions using two axes such as \overrightarrow{xyx} , there is a slight difference in the derivation. The isolated term is not s_2 but c_2 , see (A.70), and the domain of α_2 is now $[0, \pi]$, corresponding to the range of the inverse cosine function. Gimbal lock occurs for $\alpha_2 \in \{0, \pi\}$. Everything else works in the same way as described above, except that now the s_2 terms cancel out in the arguments of the inverse tangents.

Tab. A.4 gives the resulting conversion formulas for all twelve Euler conventions with left-to-right multiplication order. As usual, the formulas for the right-to-left multiplication order can be read off by looking up the formulas for the reversed rotation order and interchanging α_1 with α_3 .

A.6.4 Quaternion to Euler Angles

To convert a unit quaternion $q = (w + xi + yj + zk) \in \mathbb{H}$, $\|q\| = 1$, to Euler angles, we can compose the quaternion-to-matrix conversion (A.23) with the matrix-to-Euler conversion (Sect. A.6.3) for the desired Euler convention. Tab. A.5 gives the resulting formulas for all twelve Euler conventions with left-to-right multiplication order.

Convention	$\alpha_2 \neq \pm \frac{\pi}{2}$	$\alpha_2 = \pm \frac{\pi}{2}$
\overrightarrow{xyz}	$\alpha_1 = \arctan2(-r_{23}, r_{33})$ $\alpha_2 = \arcsin r_{13}$ $\alpha_3 = \arctan2(-r_{12}, r_{11})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin r_{13}$ $\alpha_3 = \arctan2(r_{21}, r_{22})$
\overrightarrow{xzy}	$\alpha_1 = \arctan2(r_{32}, r_{22})$ $\alpha_2 = \arcsin(-r_{12})$ $\alpha_3 = \arctan2(r_{13}, r_{11})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-r_{12})$ $\alpha_3 = \arctan2(-r_{31}, r_{33})$
\overrightarrow{yxz}	$\alpha_1 = \arctan2(r_{13}, r_{33})$ $\alpha_2 = \arcsin(-r_{23})$ $\alpha_3 = \arctan2(r_{21}, r_{22})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-r_{23})$ $\alpha_3 = \arctan2(-r_{12}, r_{11})$
\overrightarrow{yzx}	$\alpha_1 = \arctan2(-r_{31}, r_{11})$ $\alpha_2 = \arcsin r_{21}$ $\alpha_3 = \arctan2(-r_{23}, r_{22})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin r_{21}$ $\alpha_3 = \arctan2(r_{32}, r_{33})$
\overrightarrow{zxy}	$\alpha_1 = \arctan2(-r_{12}, r_{22})$ $\alpha_2 = \arcsin r_{32}$ $\alpha_3 = \arctan2(-r_{31}, r_{33})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin r_{32}$ $\alpha_3 = \arctan2(r_{13}, r_{11})$
\overrightarrow{zyx}	$\alpha_1 = \arctan2(r_{21}, r_{11})$ $\alpha_2 = \arcsin(-r_{31})$ $\alpha_3 = \arctan2(r_{32}, r_{33})$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-r_{31})$ $\alpha_3 = \arctan2(-r_{23}, r_{22})$

Convention	$\alpha_2 \notin \{0, \pi\}$	$\alpha_2 \in \{0, \pi\}$
\overrightarrow{xyx}	$\alpha_1 = \arctan2(r_{21}, -r_{31})$ $\alpha_2 = \arccos r_{11}$ $\alpha_3 = \arctan2(r_{12}, r_{13})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{11}$ $\alpha_3 = \arctan2(-r_{23}, r_{22})$
\overrightarrow{xzx}	$\alpha_1 = \arctan2(r_{31}, r_{21})$ $\alpha_2 = \arccos r_{11}$ $\alpha_3 = \arctan2(r_{13}, -r_{12})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{11}$ $\alpha_3 = \arctan2(r_{32}, r_{33})$
\overrightarrow{yxy}	$\alpha_1 = \arctan2(r_{12}, r_{32})$ $\alpha_2 = \arccos r_{22}$ $\alpha_3 = \arctan2(r_{21}, -r_{23})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{22}$ $\alpha_3 = \arctan2(r_{13}, r_{11})$
\overrightarrow{yzy}	$\alpha_1 = \arctan2(r_{32}, -r_{12})$ $\alpha_2 = \arccos r_{22}$ $\alpha_3 = \arctan2(r_{23}, r_{21})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{22}$ $\alpha_3 = \arctan2(-r_{31}, r_{33})$
\overrightarrow{zxx}	$\alpha_1 = \arctan2(r_{13}, -r_{23})$ $\alpha_2 = \arccos r_{33}$ $\alpha_3 = \arctan2(r_{31}, r_{32})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{33}$ $\alpha_3 = \arctan2(-r_{12}, r_{11})$
\overrightarrow{zyz}	$\alpha_1 = \arctan2(r_{23}, r_{13})$ $\alpha_2 = \arccos r_{33}$ $\alpha_3 = \arctan2(r_{32}, -r_{31})$	$\alpha_1 = 0$ $\alpha_2 = \arccos r_{33}$ $\alpha_3 = \arctan2(r_{21}, r_{22})$

Table A.4. Matrix-to-Euler conversion for the twelve Euler conventions with left-to-right multiplication order.

Input: rotation matrix $R = (r_{ij}) \in \text{SO}_3$ and an Euler convention.

Output: Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top$ for the respective Euler convention.

Convention	$\alpha_2 \neq \pm \frac{\pi}{2}$	$\alpha_2 = \pm \frac{\pi}{2}$
\overrightarrow{xyz}	$\alpha_1 = \arctan2(-(2yz - 2wx), 1 - 2x^2 - 2y^2)$ $\alpha_2 = \arcsin(2xz + 2wy)$ $\alpha_3 = \arctan2(-(2xy - 2wz), 1 - 2y^2 - 2z^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(2xz + 2wy)$ $\alpha_3 = \arctan2(2xy + 2wz, 1 - 2x^2 - 2z^2)$
\overrightarrow{xzy}	$\alpha_1 = \arctan2(2yz + 2wx, 1 - 2x^2 - 2z^2)$ $\alpha_2 = \arcsin(-(2xy - 2wz))$ $\alpha_3 = \arctan2(2xz + 2wy, 1 - 2y^2 - 2z^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-(2xy - 2wz))$ $\alpha_3 = \arctan2(-(2xz - 2wy), 1 - 2x^2 - 2y^2)$
\overrightarrow{yxz}	$\alpha_1 = \arctan2(2xz + 2wy, 1 - 2x^2 - 2y^2)$ $\alpha_2 = \arcsin(-(2yz - 2wx))$ $\alpha_3 = \arctan2(2xy + 2wz, 1 - 2x^2 - 2z^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-(2yz - 2wx))$ $\alpha_3 = \arctan2(-(2xy - 2wz), 1 - 2y^2 - 2z^2)$
\overrightarrow{yzx}	$\alpha_1 = \arctan2(-(2xz - 2wy), 1 - 2y^2 - 2z^2)$ $\alpha_2 = \arcsin(2xy + 2wz)$ $\alpha_3 = \arctan2(-(2yz - 2wx), 1 - 2x^2 - 2z^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(2xy + 2wz)$ $\alpha_3 = \arctan2(2yz + 2wx, 1 - 2x^2 - 2y^2)$
\overrightarrow{zxy}	$\alpha_1 = \arctan2(-(2xy - 2wz), 1 - 2x^2 - 2z^2)$ $\alpha_2 = \arcsin(2yz + 2wx)$ $\alpha_3 = \arctan2(-(2xz - 2wy), 1 - 2x^2 - 2y^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(2yz + 2wx)$ $\alpha_3 = \arctan2(2xz + 2wy, 1 - 2y^2 - 2z^2)$
\overrightarrow{zyx}	$\alpha_1 = \arctan2(2xy + 2wz, 1 - 2y^2 - 2z^2)$ $\alpha_2 = \arcsin(-(2xz - 2wy))$ $\alpha_3 = \arctan2(2yz + 2wx, 1 - 2x^2 - 2y^2)$	$\alpha_1 = 0$ $\alpha_2 = \arcsin(-(2xz - 2wy))$ $\alpha_3 = \arctan2(-(2yz - 2wx), 1 - 2x^2 - 2z^2)$

Convention	$\alpha_2 \notin \{0, \pi\}$	$\alpha_2 \in \{0, \pi\}$
\overrightarrow{xyx}	$\alpha_1 = \arctan2(2xy + 2wz, -(2xz - 2wy))$ $\alpha_2 = \arccos(1 - 2y^2 - 2z^2)$ $\alpha_3 = \arctan2(2xy - 2wz, 2xz + 2wy)$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2y^2 - 2z^2)$ $\alpha_3 = \arctan2(-(2yz - 2wx), 1 - 2x^2 - 2z^2)$
\overrightarrow{xzx}	$\alpha_1 = \arctan2(2xz - 2wy, 2xy + 2wz)$ $\alpha_2 = \arccos(1 - 2y^2 - 2z^2)$ $\alpha_3 = \arctan2(2xz + 2wy, -(2xy - 2wz))$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2y^2 - 2z^2)$ $\alpha_3 = \arctan2(2yz + 2wx, 1 - 2x^2 - 2y^2)$
\overrightarrow{yxy}	$\alpha_1 = \arctan2(2xy - 2wz, 2yz + 2wx)$ $\alpha_2 = \arccos(1 - 2x^2 - 2z^2)$ $\alpha_3 = \arctan2(2xy + 2wz, -(2yz - 2wx))$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2x^2 - 2z^2)$ $\alpha_3 = \arctan2(2xz + 2wy, 1 - 2y^2 - 2z^2)$
\overrightarrow{yzy}	$\alpha_1 = \arctan2(2yz + 2wx, -(2xy - 2wz))$ $\alpha_2 = \arccos(1 - 2x^2 - 2z^2)$ $\alpha_3 = \arctan2(2yz - 2wx, 2xy + 2wz)$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2x^2 - 2z^2)$ $\alpha_3 = \arctan2(-(2xz - 2wy), 1 - 2x^2 - 2y^2)$
\overrightarrow{zxx}	$\alpha_1 = \arctan2(2xz + 2wy, -(2yz - 2wx))$ $\alpha_2 = \arccos(1 - 2x^2 - 2y^2)$ $\alpha_3 = \arctan2(2xz - 2wy, 2yz + 2wx)$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2x^2 - 2y^2)$ $\alpha_3 = \arctan2(-(2xy - 2wz), 1 - 2y^2 - 2z^2)$
\overrightarrow{zyz}	$\alpha_1 = \arctan2(2yz - 2wx, 2xz + 2wy)$ $\alpha_2 = \arccos(1 - 2x^2 - 2y^2)$ $\alpha_3 = \arctan2(2yz + 2wx, -(2xz - 2wy))$	$\alpha_1 = 0$ $\alpha_2 = \arccos(1 - 2x^2 - 2y^2)$ $\alpha_3 = \arctan2(2xy + 2wz, 1 - 2x^2 - 2z^2)$

Table A.5. Quaternion-to-Euler conversion for the twelve Euler conventions with left-to-right multiplication order.

Input: unit quaternion $q = (w + xi + yj + zk) \in \mathbb{H}$, $\|q\| = 1$, and an Euler convention.

Output: Euler angles $(\alpha_1, \alpha_2, \alpha_3)^\top$ for the respective Euler convention.

Appendix B

Motion Smoothing with Quaternion Filters

Motion capture data is often noisy and may contain severe artifacts such as discontinuities in the trajectories, see Fig. 2.9 for an example. In Sect. 3.3.1, we have successfully applied smoothing based on 3D trajectories to suppress noise in the computation of velocity data for individual joints. We will present alternative smoothing methods in this chapter.

Other features consider joint angles or the relative position of several joints at a time. Here, it does not make sense to remove noise by smoothing all 3D trajectories that define a motion: this may violate the distance constraints imposed by the skeleton. In other words, since each 3D trajectory would be smoothed individually, the bones could suddenly become longer or shorter, see also [Krü06]. In extreme cases, this would lead to anatomically impossible joint constellations. Instead, we pursue the approach of smoothing the angle trajectories of our animated skeleton and then applying forward kinematics to obtain smoothed 3D trajectories while ensuring that the skeletal distance constraints are preserved. However, due to the spherical structure of the group of 3D rotations, smoothing angle trajectories is not as straightforward as smoothing 3D position data, see also App. A. This topic has received much attention in the literature, see, for example, [BF01, FLPJ04, LS02] and the references therein.

We evaluated three smoothing approaches using angle trajectories in quaternion representation. The first two approaches compute different kinds of weighted moving averages over quaternion sequences. The third approach, *orientation filtering*, transfers the concept of LTI filters to the set of unit quaternions, S^3 . This is done by considering displacements between successive quaternions in the tangent space to S^3 , which is isomorphic to \mathbb{R}^3 , then suitably filtering these displacements, and finally mapping the result back to S^3 . Directly smoothing the Euler angles that are usually provided in motion capture files does not make sense due to the singularities of the Euler angle parametrization, see App. A.

In the following, we assume that we want to smooth an angle trajectory represented as a sequence of unit quaternions, $q : [1 : T] \rightarrow S^3$. We use the notation $q(t)$ for $t \in [1 : T]$ to refer to individual quaternions in the sequence q . To avoid the troublesome boundary cases, we apply symmetric padding as defined in (3.24), yielding the infinite sequence $\tilde{q} : \mathbb{Z} \rightarrow S^3$.

B.1 Weighted Averages in \mathbb{R}^4

This first approach could be dubbed the “brute force” approach since it does not respect the spherical geometry of S^3 . The idea is straightforward and, according to Buss and Fillmore [BF01], has been used extensively in the literature for computing averages on spheres. Given a sequence of real weights, $h = (h_k)_{k \in [-K:K]}$ with $\sum_{k=-K}^K h_k = 1$, we regard the input quaternions as vectors in \mathbb{R}^4 and compute their convolution $(h * \tilde{q})(t)$ for $t \in [1 : T]$. As in Sect. 3.3.1, the convolution between a one-dimensional sequence and a sequence of vectors is meant in a coordinate-wise sense:

$$C^h[\tilde{q}](t) = (h * \tilde{q})(t) = \sum_{k=-K}^K h_k \tilde{q}(t - k). \quad (\text{B.1})$$

The resulting linear combinations will usually not be in S^3 , so we simply normalize them back to S^3 . This leads to the filter $\Phi_{\mathbb{R}^4}^h$, which computes the normalized, weighted average in \mathbb{R}^4 as follows, see also [LS02]:

$$\Phi_{\mathbb{R}^4}^h[\tilde{q}](t) := \begin{cases} \frac{C^h[\tilde{q}](t)}{\|C^h[\tilde{q}](t)\|} & \text{if } \|C^h[\tilde{q}](t)\| > 0 \\ \tilde{q}(t) & \text{otherwise.} \end{cases} \quad (\text{B.2})$$

Observe that the case where a linear combination has a length of exactly zero is handled by returning the quaternion $\tilde{q}(t)$ that is currently at the center of the (imagined) sliding mask. Lee and Shin [LS02] discuss this strategy of handling zero-length linear combinations. The motivation is to simulate the behavior of an averaging filter, which does not alter a symmetric configuration. Assuming for a moment the typical case of strictly positive weights, we are dealing with convex combinations. For a convex combination of unit quaternions to be zero, their linear span in \mathbb{R}^4 must include the origin. This implies that not all quaternions can lie within one (open) hemisphere, so their mutual distances must be relatively large. Since moving along half a great circle on S^3 already corresponds to a full rotation by 2π , the corresponding motion will typically be very fast.

A theoretical constellation where a convex combination would be exactly zero is the case where all weights are identical and all input quaternions are equally spaced along a great circle on S^3 . Now imagine some slight noise moving the quaternions out of their ideal configuration. Typically, the resulting convex combination would be a nonzero vector of very small magnitude pointing in a random direction. The “average” unit quaternion would then be obtained as a normalized version of this random direction. In this way, minor noise components can be strongly amplified in such unstable configurations. Generally, the normalized, weighted average in \mathbb{R}^4 will only be meaningful if the input quaternions lie within a small neighborhood, since the spherical geometry of S^3 does not play an important role in that case.

B.2 Spherical Weighted Averages

Karcher [Kar77] first studied the *intrinsic mean* of a finite set of points on a Riemannian manifold, which has later become known as the *Karcher mean*. Buss and Fillmore [BF01] reinvented the Karcher mean for the special case of *spherical averages*, compare the description in [FLPJ04]. The aim is to define a weighted average of unit quaternions purely in terms of

Algorithm B.1 SPHERICALWEIGHTEDAVERAGE

Input: $q(1), \dots, q(N) \in S^3$: Unit quaternions, all lying within an open hemisphere of S^3

$h_1, \dots, h_N \in \mathbb{R}_{>0}$: Positive weights with $\sum_{n=1}^N h_n = 1$

$\varepsilon > 0$: Termination if length of gradient is smaller than ε

Output: $q \in S^3$ approximating $A(h_1, q(1), \dots, h_N, q(N))$

Procedure:

1. Initialize $q^0 := \frac{\sum_{n=1}^N h_n q(n)}{\|\sum_{n=1}^N h_n q(n)\|}$.

2. Iterate

$$q^{j+1} := q^j \exp \left(\sum_{n=1}^N h_n \log((q^j)^{-1} q(n)) \right) \tag{B.3}$$

until $\left\| \sum_{n=1}^N h_n \log((q^j)^{-1} q(n)) \right\| < \varepsilon$ for some j .

3. Return $q := q^{j+1}$.

the intrinsic geometry of S^3 . To this end, the geodesic distance

$$\text{dist}(q_0, q_1) := \arccos \langle q_0, q_1 \rangle = \|\log_{q_0}(q_1)\| \tag{B.4}$$

of unit quaternions $q_0, q_1 \in S^3$ is considered, see also Sect. 2.5.1 as well as Sect. A.4 for the definition of the quaternionic logarithm and the quaternionic exponential. The spherical average of q_0 and q_1 is uniquely determined if $q_0 \neq -q_1$ and can be computed by spherical linear interpolation as $q_{\frac{1}{2}} := \text{slerp}(\frac{1}{2}; q_0, q_1)$, cf. Sect. A.3. Now $q_{\frac{1}{2}}$ satisfies the property that it minimizes the total squared geodesic distance $\text{dist}(q, q_0)^2 + \text{dist}(q, q_1)^2$ among all choices for q . Generalizing the total squared geodesic distance to N quaternions $q(1), \dots, q(N) \in S^3$ and associated positive weights $h_1, \dots, h_N \in \mathbb{R}_{>0}$ with $\sum_{n=1}^N h_n = 1$, we define the cost function

$$f(q; h_1, q(1), \dots, h_N, q(N)) := \sum_{n=1}^N h_n \text{dist}(q, q(n))^2 \tag{B.5}$$

and then define the *spherical weighted average* of $q(1), \dots, q(N)$ as

$$A(h_1, q(1), \dots, h_N, q(N)) := \underset{q \in S^3}{\text{argmin}} f(q; h_1, q(1), \dots, h_N, q(N)). \tag{B.6}$$

In a vector space, the centroid, $\sum_{n=1}^N h_n q(n)$, would be the unique minimizer of f , but we have seen that S^3 is not closed under weighted summation. As a further difference, the spherical weighted average does not share the associativity property of the centroid. In other words, one cannot compute $A(h_1, q(1), \dots, h_N, q(N))$ by successive slerps between point pairs. By contrast, in a vector space, the centroid can be computed by forming successive convex combinations of point pairs.

One can show that the spherical weighted average exists and is unique if all quaternions q_n lie in a common open hemisphere—which excludes the configuration that has been discussed above, where all quaternions are equally spaced along a great circle. In the non-degenerate

case, however, $A(h_1, q(1), \dots, h_N, q(N))$ can be approximated to arbitrary precision by an iterative gradient descent method with linear convergence rate, see [BF01, FLPJ04]. The authors of [BF01] also give a more elaborate algorithm with quadratic convergence rate, which we do not discuss here. The algorithm uses a result by Karcher [Kar77] stating that the negative gradient of f at $q \in S^3$ is the centroid of the logarithms of the $q(n)$ taken at q :

$$-\nabla f(q) = \sum_{n=1}^N h_n \log_q(q(n)) \quad (\text{B.7})$$

Introducing a step size $\delta > 0$, this yields the following iterative gradient descent:

$$\begin{aligned} q^{j+1} &:= \exp_{q^j} \left(\delta \sum_{n=1}^N h_n \log_{q^j}(q(n)) \right) \\ &= q^j \exp \left((q^j)^{-1} \delta \sum_{n=1}^N h_n q^j \log((q^j)^{-1} q(n)) \right) \\ &= q^j \exp \left(\delta \sum_{n=1}^N h_n \log((q^j)^{-1} q(n)) \right). \end{aligned} \quad (\text{B.8})$$

The exponential at q^j , \exp_{q^j} , maps the negative gradient back to S^3 and appends it to q^j as a spherical displacement, yielding the updated unit quaternion q^{j+1} . Buss and Fillmore [BF01] show that using a constant step size of $\delta = 1$ is sufficient for spherical data. They also suggest to use the normalized, weighted average in \mathbb{R}^4 for initialization, $q^0 := \frac{\sum_{n=1}^N h_n q(n)}{\|\sum_{n=1}^N h_n q(n)\|}$. This leads to Algorithm B.1, SPHERICALWEIGHTEDAVERAGE.

In analogy to the filter $\Phi_{\mathbb{R}^4}^h$, which computes normalized, weighted averages in \mathbb{R}^4 , we define the spherical weighted averaging filter $\Phi_{S^3}^h$ by means of

$$\Phi_{S^3}^h[\tilde{q}](t) := A(h_{-K}, \tilde{q}(t-K), \dots, h_K, \tilde{q}(t+K)) \quad (\text{B.9})$$

for $t \in [1 : T]$, where we assume positive filter coefficients $h = (h_k)_{k \in [-K:K]}$ summing to 1. As we will see in Sect. B.4, the operators $\Phi_{\mathbb{R}^4}^h$ and $\Phi_{S^3}^h$ yield very similar outputs most of the time; only for quaternion signals exhibiting large angular velocities, they behave differently, see also the discussion in Sect. B.1. Even though the spherical averaging filter, $\Phi_{S^3}^h$, provides a sound concept of averaging on S^3 , one could raise the question of whether the overhead of spherical averaging is justified in our application of motion smoothing—the degenerate cases where $\Phi_{\mathbb{R}^4}^h$ fails rarely occur in motion data.

B.3 Orientation Filters

Lee and Shin [LS02] introduce *orientation filters* to transfer the concept of LTI filters to S^3 . They consider the general case of (possibly negative) filter coefficients $h = (h_k)_{k \in [-K:K]}$ with $\sum_{k=-K}^K h_k = 1$, and pursue a three-step strategy: first, compute displacements

$$\omega(t) := \log(q^{-1}(t)q(t+1)) \in \mathbb{R}^3 \quad (\text{B.12})$$

for $t \in [1 : T-1]$, which can be interpreted as angular velocities. Then convolve the padded sequence $\tilde{\omega}$ with a filter mask g that can be precomputed from h , yielding as output a *filter*

Algorithm B.2 ORIENTATIONFILTER

Input: $q : [1 : T] \rightarrow S^3$: Quaternion trajectory with $\text{dist}(q(t), q(t+1)) < \frac{\pi}{2}$ (see Sect. A.3)

$h_{-K}, \dots, h_K \in \mathbb{R}$: Arbitrary weights with $\sum_{k=-K}^K h_k = 1$

Output: $\Phi_o^h[q] : [1 : T] \rightarrow S^3$, the filtered version of q

Procedure:

1. Compute the orientation filter mask $g = g(h)$ as in (B.17).
2. Compute the sequence of angular velocities, $\omega : [1 : T - 1] \rightarrow \mathbb{R}^3$, as

$$\omega(t) := \log(q^{-1}(t)q(t+1)) \quad (\text{B.10})$$

3. Compute the filtered quaternion sequence from the padded version of ω as

$$\Phi_o^h[q](t) := q(t) \exp\left(\sum_{k=-K}^{K-1} g_k \tilde{\omega}(t-k)\right). \quad (\text{B.11})$$

gain vector in \mathbb{R}^3 for each time index t . Finally, apply the filter gain at time t to the corresponding original data point, $q(t)$, by means of the exponential map.

The underlying idea is to view each quaternion of the input sequence as a multiplicative cumulation of its predecessors by means of the telescoping product

$$q(t) = q(1) \prod_{k=1}^{t-1} q^{-1}(k)q(k+1) = q(1) \prod_{k=1}^{t-1} \exp(\omega(k)). \quad (\text{B.13})$$

From this, an additive cumulative representation for q in terms of a vector sequence $p : [1 : T] \rightarrow \mathbb{R}^3$ is derived by demanding $p(t+1) - p(t) = \log(q^{-1}(t)q(t+1)) = \omega(t)$, where the starting point $p(1) \in \mathbb{R}^3$ can be chosen arbitrarily:

$$p(t) := p(1) + \sum_{k=1}^{t-1} \log(q^{-1}(k)q(k+1)). \quad (\text{B.14})$$

If $q(1)$ is known, the sequence q can be reconstructed from p by evaluating

$$q(t) = q(1) \prod_{k=1}^{t-1} \exp(p(k+1) - p(k)). \quad (\text{B.15})$$

The sequence p can be interpreted as a sampled curve in \mathbb{R}^3 , the linear velocity profile of which coincides with the angular velocity profile of q . In particular, note that both linear and angular velocities are vectors in \mathbb{R}^3 . We will see that p need not be computed explicitly, but it helps to build intuition.

Applying the convolution mask h to the symmetrically padded vector sequence \tilde{p} , one obtains the filtered sequence $C^h[\tilde{p}]$. Instead of directly mapping $C^h[\tilde{p}]$ back to S^3 via (B.15), which would introduce drifting errors, Lee and Shin [LS02] consider the *filter gains* $C^h[\tilde{p}](t) -$

$p(t)$, which are vectors in \mathbb{R}^3 . The exponentials of the filter gains are spherical displacements in S^3 of the same magnitude as the filter gains (since the exponential map preserves lengths). These spherical displacements are applied to the original data points, $q(t)$, by means of quaternion multiplication. This yields the *orientation filter*

$$\Phi_o^h[q](t) := q(t) \exp\left(C^h[\tilde{p}](t) - p(t)\right), \quad (\text{B.16})$$

where the ‘o’ stands for “orientation”.

It turns out that the cumulative definition of p allows us to compute the filter gains from the sequence ω without explicitly considering the sequence p . One can check that the following *orientation filter mask* g constructed from h by

$$g_k := \begin{cases} \sum_{j=k+1}^K a_j & \text{if } 0 \leq k \leq K-1 \\ -\sum_{j=-K}^k a_j & \text{if } -K \leq k < 0 \end{cases} \quad (\text{B.17})$$

satisfies the identity $C^g[\tilde{\omega}](t) = C^h[\tilde{p}](t) - p(t)$ for $t \in [1 : T]$, see [LS02]. Algorithm B.2 summarizes the computation of this filtering process.

Discussion. Orientation filters inherit important properties from LTI (linear time invariant) filters for vector space data, see [LS02]. They are time invariant as well as invariant under coordinate transformations: in a pointwise sense, $a \Phi_o^h[q] b = \Phi_o^h[aqb]$ for all $a, b \in S^3$ and $q : [1 : T] \rightarrow S^3$. As an advantage over spherical weighted average filters $\Phi_{S^3}^h$, orientation filters can handle arbitrary filter coefficients. This also enables highpass filtering of orientation data, which typically requires negative filter coefficients. Highpass filters for motion data can be used for multiresolution analysis to extract the motion texture [Pul02, PB02].

In case we assume the filter coefficients to be strictly positive, orientation filters are remarkably similar to spherical weighted averages for a wide range of input data. We will now compare the two approaches to explain the similarity of filter outputs that we observed in our experiments, see Sect. B.4. Plugging the expression (B.10) for $\omega(t)$ into (B.11), we obtain (up to symmetric padding)

$$\Phi_o^h[q](t) = q(t) \exp\left(\sum_{k=-K}^K g_k \log(q^{-1}(t-k)q(t-k+1))\right). \quad (\text{B.18})$$

Formally, the differences between (B.18) and the iteration formula (B.3) for the spherical weighted average are that the inverse quaternion appearing in the logarithms in (B.18) is not constant for each term of the sum, as in (B.3), and the filter mask g is used instead of h . As an illustration, consider Fig. B.1. Given seven unit quaternions $q(-3), \dots, q(3) \in S^3$ and the filter mask $h = (\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7})$, we compute iteration number $j+1$ of the spherical averaging algorithm (top row of Fig. B.1) as well as the filter gain for the corresponding orientation filter mask, $g = (-\frac{1}{7}, -\frac{2}{7}, -\frac{3}{7}, \frac{3}{7}, \frac{2}{7}, \frac{1}{7})$ (bottom row of Fig. B.1). To make the two situations comparable, we assume that the current estimate of the spherical weighted average is $q^j = q(0)$, which has actually been proposed by Fletcher [FLPJ04] as an alternative mode of initialization (q^0) for the spherical weighted average. The next step is to compute the logarithms of certain spherical displacements: for the spherical weighted average, the displacements are taken from $q(0)$ to the other six quaternions, while for the orientation filter, the displacements are taken between successive quaternions.

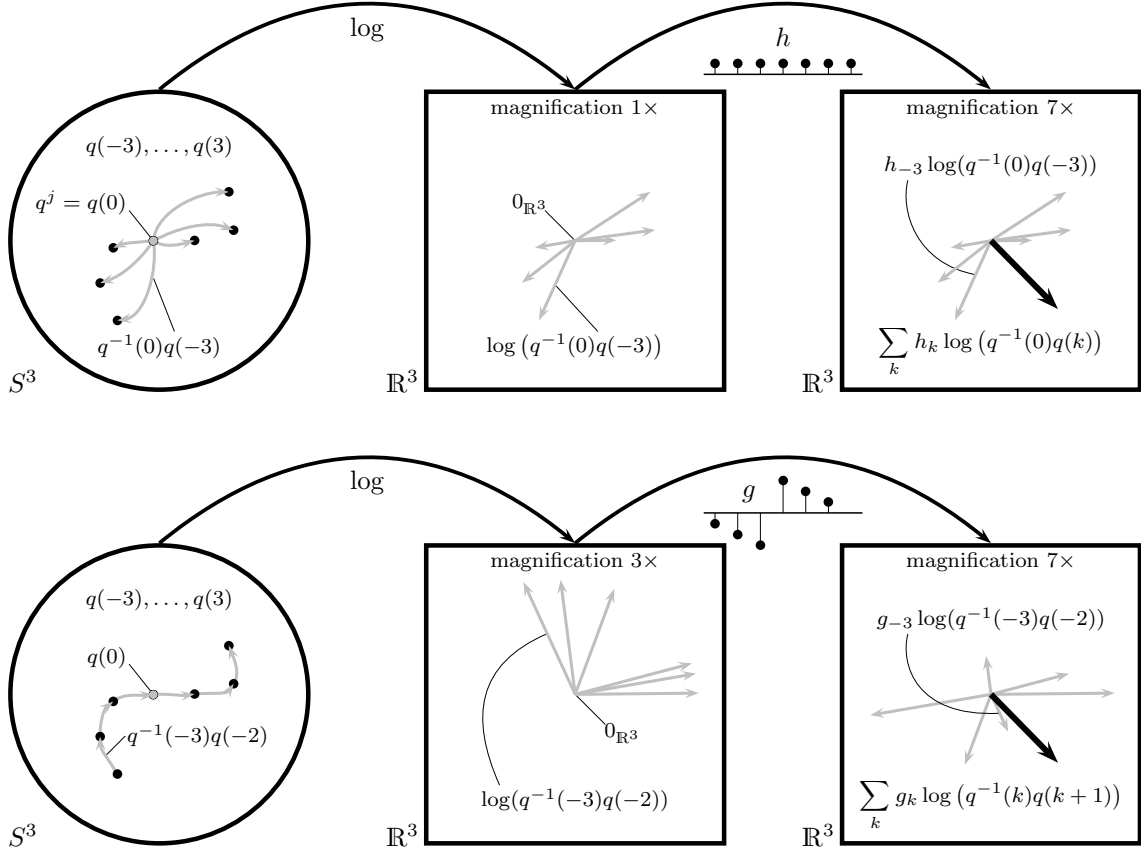


Figure B.1. Top row: Computing the gradient for spherical averaging with $h = (\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7})$. **Bottom row:** Computing a filter gain with the orientation filter mask $g = (-\frac{1}{7}, -\frac{2}{7}, -\frac{3}{7}, \frac{3}{7}, \frac{2}{7}, \frac{1}{7})$. The left column shows a point set on S^3 with spherical displacements shown as gray arrows. The middle column shows the logarithms of these displacements. The right column shows the respective weighted versions of the logarithms together with their vector sums, which are drawn as bold arrows.

The resulting logarithms are vectors in \mathbb{R}^3 . For the case of spherical averaging, their weighted sum with respect to h is the desired negative gradient, $-\nabla f$, of the cost function, f . For the case of orientation filtering, the special construction of g has the effect that displacements for $k < 0$ are weighted with negative coefficients, while displacements for $k \geq 0$ are weighted with positive coefficients. By this construction, the resulting filter gain computed by the orientation filter is very similar to the negative gradient computed by the iterative spherical averaging. In fact, if we were not dealing with spherical data but with vector space data, the two resulting vectors would be exactly the same, see [LS02]. In our case, the closeness of the two results depends on the diameter of the set of input quaternions. The smaller the diameter, the less the distortion that is incurred by the logarithm map to the tangent space and the exponential map back to S^3 . A small diameter will typically be given if the underlying motion is not too fast and the filter mask is not too long. Under this assumption, the vector displacements computed by spherical averaging and orientation filtering, respectively, are very similar. Both algorithms then append the respective displacements to the quaternion $q(0)$ by means of the exponential map. Thus, under certain conditions, one iteration of spherical averaging has a similar effect as orientation filtering.

We note a further important property of orientation filters: they can handle degenerate

cases such as symmetric configurations along a great circle, which are invalid as input for spherical weighted average filters. This is due to the fact that orientation filters compute spherical displacements locally, between successive quaternions. Therefore, the logarithms are always defined as long as the geodesic distances between successive quaternions are guaranteed to be less than π .

B.4 Evaluation of Smoothing Strategies

Objective evaluation of filter methods for orientation data is difficult due to the lack of analysis techniques such as the Fourier transform for time-discrete signals $q : [1 : T] \rightarrow S^3$ or time-continuous signals $q : \mathbb{R} \rightarrow S^3$. We will therefore discuss some representative examples demonstrating the noise removal properties of the three filtering strategies. We start with a synthetic example, see Fig. B.2. The signal q consists of two repetitions of a smooth path on S^3 and is artificially degraded in the following way. We generated a 3D zero-mean Gaussian white noise signal with $\sigma = 0.1$, computed the corresponding spherical displacements via the exponential map and applied these displacements to the quaternions in the signal q , yielding the noisy signal q' . Around frame 75, a discontinuity was inserted by setting five successive frames to some constant quaternion (not an antipodal jump.) Also note the discontinuity in the middle of the signal that results from the concatenation of the two repetitions.

We used the Gaussian filter mask h with a relatively large length of $2K + 1 = 71$, see Eqn. (3.25), which is very similar to the binomial filter mask proposed by Lee and Shin [LS02]—actually, the binomial filter mask is an approximation of the Gaussian filter mask. The three filters $\Phi_{\mathbb{R}^4}^h$, $\Phi_{S^3}^h$, and Φ_o^h were then applied to q' , taking 60 ms, 6,600 ms, and 125 ms to compute, respectively. The former two filters succeeded in removing most of the Gaussian noise. The orientation filter, Φ_o^h , yielded a similar output as the spherical average filter, $\Phi_{S^3}^h$, but the noise suppression was not as successful, especially during the discontinuities (where high angular velocities occur). This effect is most likely due to the distortions incurred by the logarithm map—the filter gains computed in \mathbb{R}^3 do not exactly correspond to the ideal displacements that would be required in S^3 . Comparing $\Phi_{\mathbb{R}^4}^h$ and $\Phi_{S^3}^h$, there are almost no differences in the filter output except at the discontinuity at $t = 200$, where $\Phi_{\mathbb{R}^4}^h$ produced some distortions opposed to the “ideal average” computed by $\Phi_{S^3}^h$.

Regarding real motion capture data, we investigated the ballet motion of Fig. 2.9. The angle trajectory of the joint ‘lankle’ is slightly noisy and exhibits a discontinuity at frame 130 that results in a visible flip of the ankle, see Fig. B.3 and Fig. B.4. Again, all three filtering strategies yielded visually very similar results and succeeded in removing most of the noise. The similarity of the outputs can also be seen in Fig. B.5, where we extend the velocity example of Fig. 3.12. After filtering all joints’ angle trajectories with the three filtering strategies, we recomputed the joints’ 3D positions via forward kinematics. Then, we computed the absolute 3D velocity as in Sect. 3.3.1. The resulting smoothed velocity curves are fairly similar to the results obtained after lowpass filtering the 3D trajectories.

As a result, we note that even though orientation filtering provides a flexible framework for applying the concept of FIR filters to S^3 , the smoothing task seems to be handled better by the other filtering strategies. For well-conditioned data, the efficient normalized weighted average filter in \mathbb{R}^4 is feasible. Spherical weighted averaging seems to yield the best results but is very slow. None of the three filtering techniques facilitates a clean removal of discontinuities. Here, either manual intervention or a “spherical median filter” would be required.

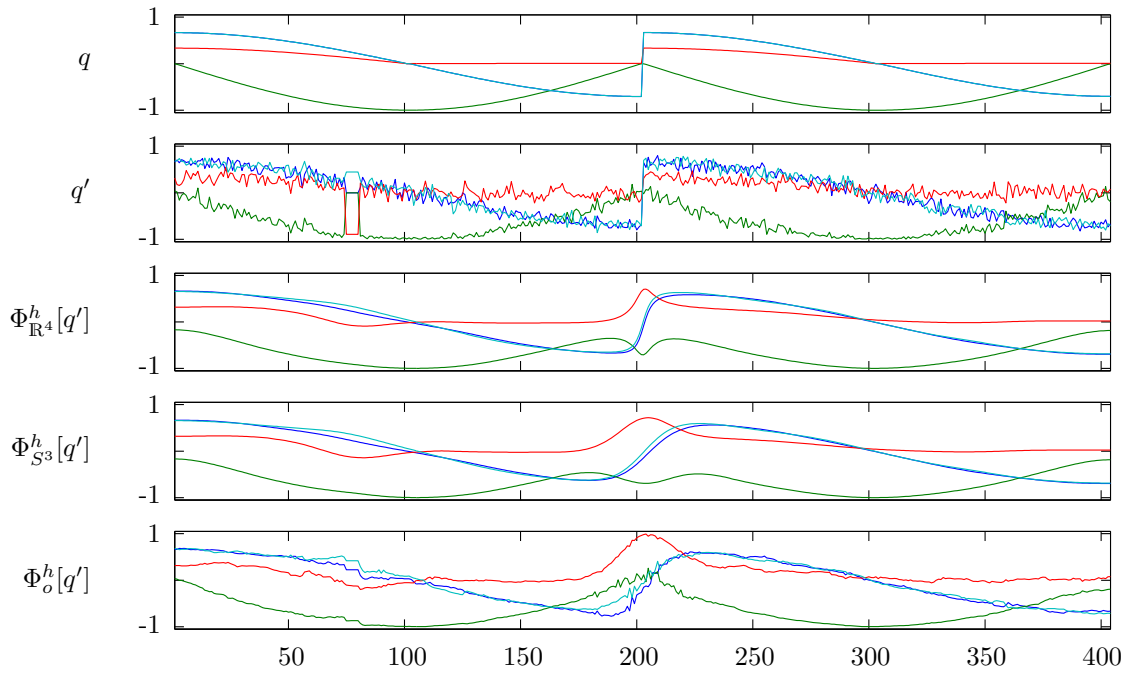


Figure B.2. Filtering a synthetic 400-frame quaternion signal with a Gaussian filter h of length 71. From top to bottom: original signal q ; added Gaussian noise ($\sigma = 0.1$); filter response of normalized averaging filter in \mathbb{R}^4 ; filter response of spherical weighted averaging filter; filter response of orientation filter. The four components of the quaternions are color-coded: blue (w), red (x), green (y), cyan (z).

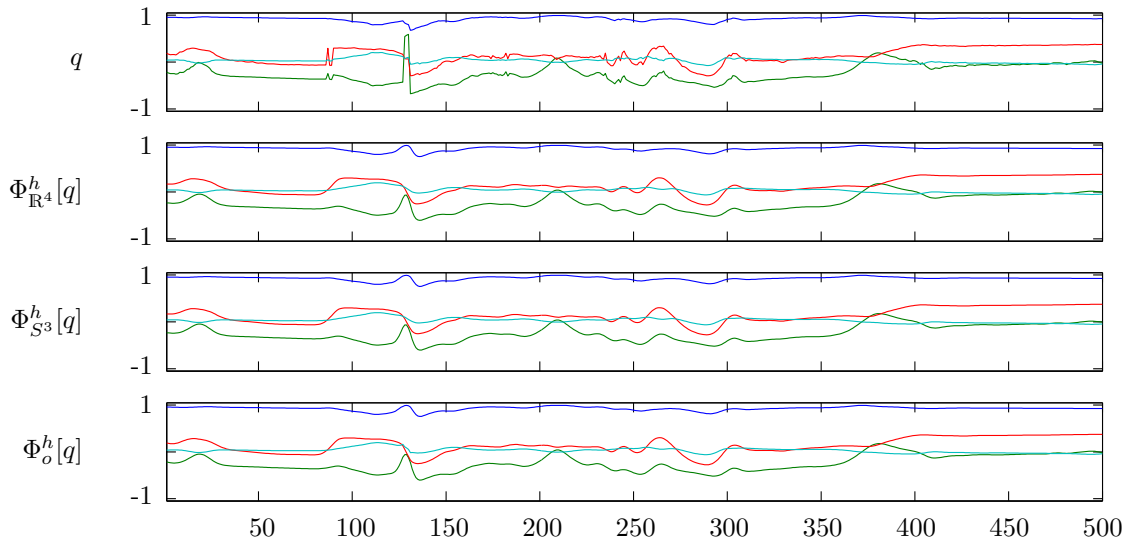


Figure B.3. Filtering a 100-frame quaternion signal corresponding to frames 100–200 of the angle trajectory of the joint ‘lankle’ in the ballet motion of Fig. 2.9. We used a Gaussian filter h of length 13. From top to bottom: original signal q ; filter response of normalized averaging filter in \mathbb{R}^4 ; filter response of spherical weighted averaging filter; filter response of orientation filter. The four components of the quaternions are color-coded: blue (w), red (x), green (y), cyan (z).

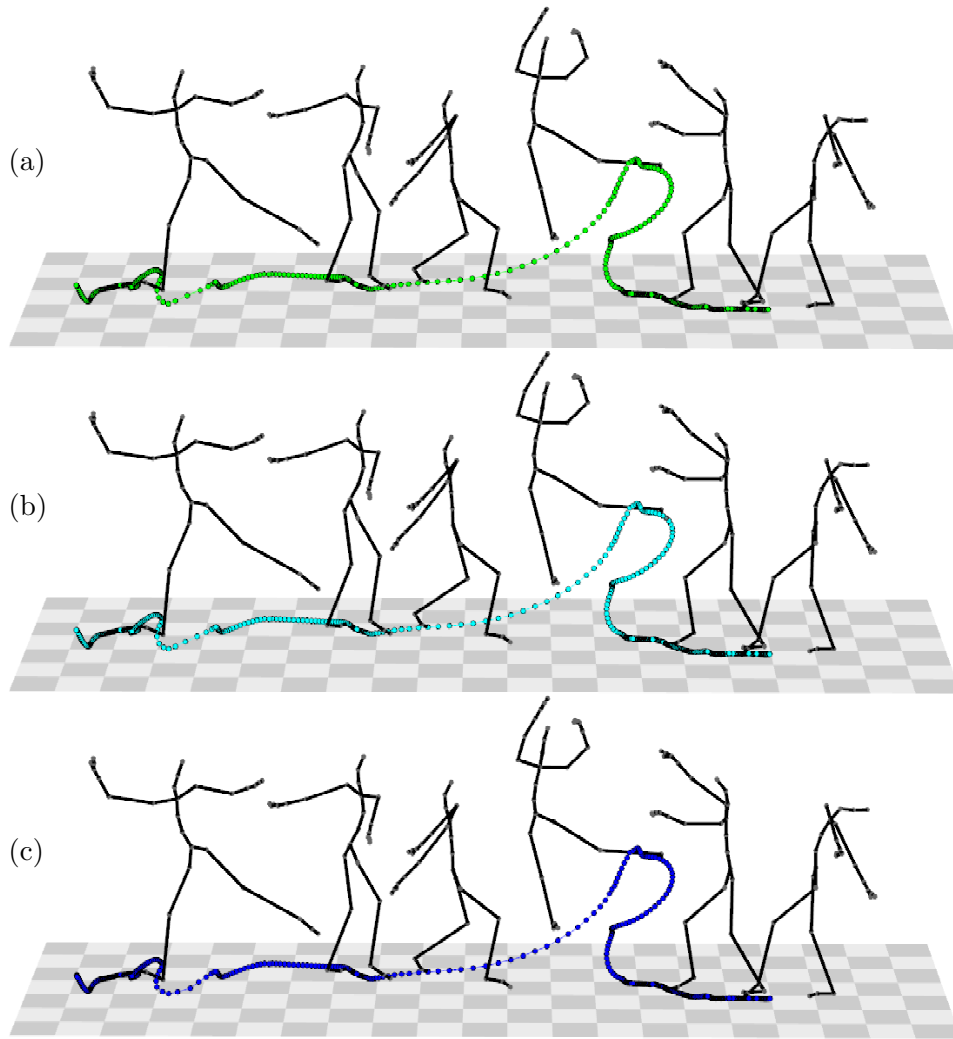


Figure B.4. The 500-frame ballet motion of Fig. 1.7 and Fig. 2.9 filtered with (a) normalized averaging in \mathbb{R}^4 , (b) spherical weighted averaging, (c) orientation filtering using a Gaussian filter of length 13, or 108 ms, for all angle trajectories. The trajectory of the joint ‘toes’ is shown.

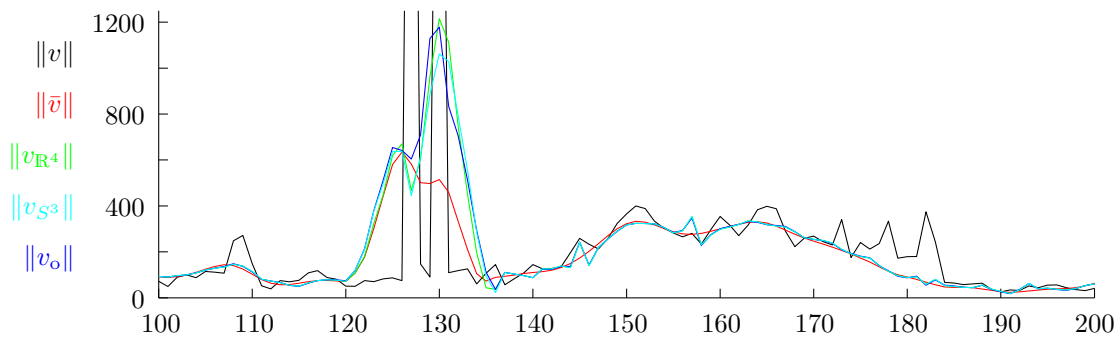


Figure B.5. Detail (frames 100–200) of the velocity curves of Fig. 3.12, plus the velocity curves corresponding to the filtered ballet motions of Fig. B.4. $\|v\|$, black: raw 3D data; $\|\bar{v}\|$, red: after 3D filtering with a 13-frame Gaussian filter h ; $\|v_{\mathbb{R}^4}\|$, green: after normalized averaging in \mathbb{R}^4 with h ; $\|v_{S^3}\|$, cyan: after spherical weighted averaging with h ; $\|v_o\|$, blue: after orientation filtering with h .

Appendix C

2D Alignment of 3D Point Clouds

We denote the optimal alignment parameters minimizing (2.8) by $\hat{\alpha}$, \hat{x} , and \hat{z} . It turns out that these parameters can be computed analytically. The key idea is to separate the optimization of the rotation angle α from the optimization of the translational offsets x and z , which is possible due to the following lemma, cf. [AHB87, HP03, Ume91]:

Lemma C.1 *Let $(\hat{\alpha}, \hat{x}, \hat{z}) = \operatorname{argmin}_{\alpha, x, z} f(\alpha, x, z)$. Then the optimal offset parameters depend on the optimal rotation angle in the following way:*

$$\hat{p} := \begin{pmatrix} \hat{x} \\ 0 \\ \hat{z} \end{pmatrix} = \begin{pmatrix} \bar{x} - \bar{x}' \cos \hat{\alpha} - \bar{z}' \sin \hat{\alpha} \\ 0 \\ \bar{z} + \bar{x}' \sin \hat{\alpha} - \bar{z}' \cos \hat{\alpha} \end{pmatrix} = \bar{p} - R_y(\hat{\alpha})\bar{p}' \quad (\text{C.1})$$

where $\bar{p} := (\bar{x}, 0, \bar{z})^\top$ is the projection of the centroid of H onto the xz plane, $\bar{x} := \sum_{k=1}^K w_k x_k$ is the x component of \bar{p} , and the other barred terms are defined similarly.

Proof: According to the following rule for differentiable mappings $F : \mathbb{R}^a \rightarrow \mathbb{R}^b$,

$$\frac{\partial}{\partial u_i} \|F(u)\|^2 = 2 \sum_{j=1}^b F_j(u) \frac{\partial}{\partial u_i} F_j(u), \quad (\text{C.2})$$

the partial derivative of f with respect to x is

$$\begin{aligned} \frac{\partial f(\alpha, x, z)}{\partial x} &= -2 \sum_{k=1}^K w_k (x_k - x'_k \cos \alpha - z'_k \sin \alpha - x) \\ &= -2(\bar{x} - \bar{x}' \cos \alpha - \bar{z}' \sin \alpha - x), \end{aligned} \quad (\text{C.3})$$

where we exploited that the w_k sum to unity. Similarly, the partial derivative of f with respect to z is

$$\frac{\partial f(\alpha, x, z)}{\partial z} = -2(\bar{z} + \bar{x}' \sin \alpha - \bar{z}' \cos \alpha - z). \quad (\text{C.4})$$

Since the parameters $\hat{\alpha}$, \hat{x} , and \hat{z} minimize the error function f , the derivatives (C.3) and (C.4) must vanish at $(\hat{\alpha}, \hat{x}, \hat{z})$. Setting $\left. \frac{\partial f(\alpha, x, z)}{\partial x} \right|_{\hat{\alpha}, \hat{x}, \hat{z}}$ and $\left. \frac{\partial f(\alpha, x, z)}{\partial z} \right|_{\hat{\alpha}, \hat{x}, \hat{z}}$ to zero and solving for \hat{x} and \hat{z} , respectively, we obtain (C.1). Since the zero of the first derivative is unique and since

f is a nonnegative function, only a local minimum can occur at the resulting \hat{p} . Furthermore, \hat{p} must be the global minimizer since the domain of f is not bounded. \square

Corollary C.2 *Let $(\hat{\alpha}, \hat{x}, \hat{z}) = \operatorname{argmin}_{\alpha, x, z} f(\alpha, x, z)$. Then*

$$\hat{\alpha} = \operatorname{argmin}_{\alpha} g(\alpha) := \operatorname{argmin}_{\alpha} \sum_{k=1}^K w_k \|q_k - R_y(\alpha)q'_k\|^2, \quad (\text{C.5})$$

where $q_k := p_k - \bar{p}$, $q'_k := p'_k - \bar{p}'$ are the centralized versions of the points p_k , p'_k , respectively.

Proof: According to Lemma C.1, optimal values of x and z must obey equation (C.1). We may therefore restrict the parameter space of the original optimization problem to such offset vectors that satisfy $p = p(\alpha) = \bar{p} - R_y(\alpha)\bar{p}'$, where $p(\alpha) := (x(\alpha), 0, z(\alpha))^\top$. Plugging this expression into (2.8), we obtain

$$\begin{aligned} f(\alpha, x(\alpha), z(\alpha)) &= \sum_{k=1}^K w_k \|p_k - (R_y(\alpha)p'_k + p(\alpha))\|^2 \\ &= \sum_{k=1}^K w_k \|p_k - (R_y(\alpha)p'_k + \bar{p} - R_y(\alpha)\bar{p}')\|^2 \\ &= \sum_{k=1}^K w_k \|p_k - \bar{p} - R_y(\alpha)(p'_k - \bar{p}')\|^2 \\ &= \sum_{k=1}^K w_k \|q_k - R_y(\alpha)q'_k\|^2 \\ &= g(\alpha), \end{aligned} \quad (\text{C.6})$$

which shows that the optimal α for f can be found by minimizing g . \square

Theorem C.3 *The optimal parameters minimizing (2.8) are*

$$\hat{\alpha} = \arctan \left(\frac{\sum_{k=1}^K w_k (x_k z'_k - z_k x'_k) - (\bar{x} \bar{z}' - \bar{z} \bar{x}')}{\sum_{k=1}^K w_k (x_k x'_k + z_k z'_k) - (\bar{x} \bar{x}' + \bar{z} \bar{z}')} \right) \quad (\text{C.7})$$

$$\hat{x} = \bar{x} - \bar{x}' \cos \hat{\alpha} - \bar{z}' \sin \hat{\alpha} \quad (\text{C.8})$$

$$\hat{z} = \bar{z} + \bar{x}' \sin \hat{\alpha} - \bar{z}' \cos \hat{\alpha} \quad (\text{C.9})$$

Proof: The optimal offset values, \hat{x} and \hat{z} , have been derived in Lemma C.1. According to Corollary C.2, we can minimize g to determine the optimal rotation angle, $\hat{\alpha}$. Introducing

the symbols $q_k =: (X_k, Y_k, Z_k)^\top$ and $q'_k =: (X'_k, Y'_k, Z'_k)^\top$, the derivative of g is

$$\begin{aligned}
\frac{dg(\alpha)}{d\alpha} &\stackrel{(C.2)}{=} 2 \sum_{k=1}^K w_k [(X_k - X'_k \cos \alpha - Z'_k \sin \alpha)(X'_k \sin \alpha - Z'_k \cos \alpha) \\
&\quad + (Z_k + X'_k \sin \alpha - Z'_k \cos \alpha)(X'_k \cos \alpha + Z'_k \sin \alpha)] \\
&= 2 \sum_{k=1}^K w_k [(X_k X'_k + Z_k Z'_k) \sin \alpha + (-X_k Z'_k + Z_k X'_k) \cos \alpha \\
&\quad + (-(X'_k)^2 + (Z'_k)^2 + (X'_k)^2 - (Z'_k)^2) \sin \alpha \cos \alpha \\
&\quad + X'_k Z'_k (\sin^2 \alpha + \cos^2 \alpha) - X'_k Z'_k (\sin^2 \alpha + \cos^2 \alpha)] \\
&= 2 \sum_{k=1}^K w_k [(X_k X'_k + Z_k Z'_k) \sin \alpha - (X_k Z'_k - Z_k X'_k) \cos \alpha].
\end{aligned} \tag{C.10}$$

We introduce the abbreviations

$$A := \sum_{k=1}^K w_k (X_k X'_k + Z_k Z'_k) \tag{C.11}$$

$$B := \sum_{k=1}^K w_k (X_k Z'_k - Z_k X'_k) \tag{C.12}$$

and solve the equation $\left. \frac{dg(\alpha)}{d\alpha} \right|_{\hat{\alpha}} = 0$ for $\hat{\alpha}$, yielding

$$\begin{aligned}
&A \sin \hat{\alpha} = B \cos \hat{\alpha} \\
\Leftrightarrow &\frac{\sin \hat{\alpha}}{\cos \hat{\alpha}} = \frac{B}{A} \\
\Leftrightarrow &\hat{\alpha}_1 = \arctan \left(\frac{B}{A} \right) \quad \vee \quad \hat{\alpha}_2 = \arctan \left(\frac{B}{A} \right) + \pi,
\end{aligned} \tag{C.13}$$

where we assume the range of the arctan function to be $[0, \pi)$. Obviously, the two possible solutions $\hat{\alpha}_{1,2}$ lie on opposing sides of the unit circle. According to the extreme value theorem, g must assume a global minimum and a global maximum on its domain $\mathbb{R}/2\pi\mathbb{Z}$, since this set is compact and g is continuous. Furthermore, since $\mathbb{R}/2\pi\mathbb{Z}$ has no boundary points due to its circular topology, the global minimum must occur at one of the stationary points, $\hat{\alpha}_1$ or $\hat{\alpha}_2$. We consider the second derivative of g :

$$\frac{d^2 g(\alpha)}{d\alpha^2} = 2(A \cos \alpha + B \sin \alpha). \tag{C.14}$$

Evaluating $\frac{d^2 g(\alpha)}{d\alpha^2}$ at $\hat{\alpha}_1 \in [0, \pi)$, we obtain

$$2(A \cos \hat{\alpha}_1 + B \sin \hat{\alpha}_1) = 2c(A^2 + B^2) \geq 0, \tag{C.15}$$

since $\cos \arctan(\frac{B}{A}) = cA$ and $\sin \arctan(\frac{B}{A}) = cB$ for some $c > 0$. The degenerate case $A^2 + B^2 = 0$ only occurs when all of the points q_k and q'_k lie on the y axis and are therefore unaffected by the rotation $R_y(\alpha)$. As a sanity check, we also evaluate the second derivative at $\hat{\alpha}_2 = \hat{\alpha}_1 + \pi$, which gives

$$2(A \cos \hat{\alpha}_2 + B \sin \hat{\alpha}_2) = -2c(A^2 + B^2) \leq 0, \tag{C.16}$$

since $\sin(\hat{\alpha}_1 + \pi) = -\sin(\hat{\alpha}_1)$ and $\cos(\hat{\alpha}_1 + \pi) = -\cos(\hat{\alpha}_1)$. Therefore, the global minimum occurs at $\hat{\alpha}_1$ and the global maximum occurs at $\hat{\alpha}_2$.

Resubstituting A , B , $X_k = x_k - \bar{x}$, $Z_k = z_k - \bar{z}$, $X'_k = x'_k - \bar{x}'$, and $Z'_k = z'_k - \bar{z}'$ yields

$$\hat{\alpha} = \arctan \left(\frac{\sum_{k=1}^K w_k (x_k z'_k - z_k x'_k) - (\bar{x} \bar{z}' - \bar{z} \bar{x}')}{\sum_{k=1}^K w_k (x_k x'_k + z_k z'_k) - (\bar{x} \bar{x}' + \bar{z} \bar{z}')} \right). \quad (\text{C.17})$$

□

Appendix D

Motion Capture File Formats

D.1 The ASF/AMC Format

ASF/AMC is a skeleton-based mocap file format that was developed by the computer game producer Acclaim. With the demise of Acclaim in 2004, usage of this format seems to have been discontinued, and it is not being developed any further. Furthermore, the format is very poorly documented, the only sources are web pages such as [oW99a]. Yet there is a large corpus of ASF/AMC mocap data available to the public, for example the CMU mocap database [CMU03]. Commercial mocap software such as Vicon BodyBuilder offers export options to the ASF/AMC format.

Mocap data in ASF/AMC format is described by two separate ASCII-coded files: an ASF file contains the fixed skeleton information, while an AMC file encodes the free parameters. Typically, there will be a single ASF file for each actor, which can be used with multiple AMC files recorded by that actor. ASF files are bone-based in contrast to the joint-based BVH files (Sect. D.2). The Euler conventions used in ASF and AMC files are always based on a fixed reference frame, for example \overline{xyz} , see Sect. A.5.1. In this section, we discuss the ASF and the AMC file formats and describe how they map to animated skeletons as introduced in Sect. 1.3.3.

D.1.1 ASF

The following excerpt from an ASF file was taken from the CMU database [CMU03]. Individual sections within ASF files are delimited by keywords preceded by a colon (:), for example `:name`. An ASF file is divided into three blocks: a header, the bone data, and the skeletal hierarchy.

```
# Example of an ASF file.
# Comments are denoted by a hash sign.
:version      1.10
:name         MY_SKELETON
:units
  mass        1.0
  length      0.45
  angle       deg
:documentation
This is an ASF test file. Documentation
can have an arbitrary number of lines.
#####
:root
```

```

order      TX TY TZ  RX RY RZ
axis       XYZ
position   0      0      0
orientation 0      0      0
:bonedata
begin
  id       1
  name     lhipjoint
  direction 0.603808 -0.713975 0.35448
  length   2.2025
  axis     0      0      0      XYZ
end
begin
  id       2
  name     lfemur
  direction 0.34202 -0.939693 0
  length   6.55877
  axis     0      0      20      XYZ
  dof      RX      RY      RZ
  limits   (-160.0 20.0)
           (-70.0 70.0)
           (-60.0 70.0)
end
begin
  id       3
  name     ltibia
  direction 0.34202 -0.939693 0
  length   6.80302
  axis     0      0      20      XYZ
  dof      RX
  limits   (-10.0 170.0)
end
begin
  id       4
  name     lfoot
  direction 0.09185 -0.25235 0.963267
  length   2.03446
  axis     -90     0      20      XYZ
  dof      RX      RZ
  limits   (-45.0 90.0)
           (-70.0 20.0)
end
.
.
.
begin
  id       30
  name     rthumb
  direction -0.707107 0      0.707107
  length   0.691594
  axis     -90     -45     0      XYZ
  dof      RX      RZ
  limits   (-45.0 45.0)
           (-45.0 45.0)
end
#####
:hierarchy
begin
  root lhipjoint rhipjoint lowerback
  lhipjoint lfemur
  lfemur ltibia
  ltibia lfoot
  lfoot ltoes
  rhipjoint rfemur
  rfemur rtibia
  rtibia rfoot

```



```

rfoot rtoes
lowerback upperback
upperback thorax
thorax lowerneck lclavicle rclavicle
lowerneck upperneck
upperneck head
lclavicle lhumerus
lhumerus lradius
lradius lwrist
lwrist lhand lthumb
lhand lfingers
rclavicle rhumerus
rhumerus rradius
rradius rwrist
rwrist rhand rthumb
rhand rfingers
end

```

In the following, we explain some of the important sections of an ASF file.

:units **length** specifies a constant by which all coordinates and lengths appearing in ASF and AMC files have to be *divided* to obtain inches. This is the situation for CMU data. Other variants have been encountered where the data has to be multiplied by the constant to obtain centimeters.

angle can be either **deg** or **rad**, standing for degree or radians.

:root The root node is treated separately in ASF files since it is not a proper bone consisting of both a proximal and a distal joint, but rather a single joint. However, those bones that are directly incident to the root are not necessarily rigidly connected to the root but can move by means of their proximal joint.

order specifies the order in which the root's degrees of freedom appear in associated AMC files. For example, "TX" stands for translation in x direction, while "RY" stands for rotation about the y axis.

axis defines the Euler convention for the root coordinate system, which is independent of the order of appearance in the AMC file as specified by **order**. A value of "XYZ" stands for the Euler convention \overleftarrow{xyz} .

position is a coordinate triplet describing a translational offset for the root. This can be used to change the starting position of the skeleton without altering the AMC data.

orientation is an Euler angle triplet in the convention given by **axis**, which describes a rotational offset for the root. This can be used to change the starting orientation of the skeleton without altering the AMC data.

:bonedata A list of bones, each of which is delimited by **begin/end** pairs.

id is an optional numeric identifier for the bone.

name is supposed to contain no whitespace characters and provides a unique textual description for the bone.

direction is a coordinate triplet specifying a vector in the world coordinate system, which indicates the direction of the bone in the skeleton's neutral pose.

length is the bone's length. Multiplying a normalized version of the **direction** vector by **length**, one obtains the offset vector to the bone's distal joint.

axis is a triplet of Euler angles referring to the skeleton's neutral pose. It specifies the rotational offset of the bone's local coordinate system against the world coordinate system, always using the \overline{xyz} convention. Confusingly, the Euler convention that is given behind the **axis** Euler triplet has nothing to do with this rotational offset. Instead, it specifies the Euler convention that is used for the rotational degrees of freedom of this bone.

dof is an optional field declaring the bone's degrees of freedom and their sequence of appearance in associated AMC files, as for the case of the root's **order** field. Note that the Euler convention for the bone's local coordinate system is independent of their order of appearance in the AMC file. The Euler convention is specified by the **axis** field (see above). Usually, only rotational DOFs are specified for the bones. If less than three rotational DOFs are given, the unspecified Euler angles must be set to be zero.

limits is an optional field declaring a valid interval for each DOF, given in the same sequence as the DOFs. The intervals are specified by a start and end value and are enclosed in parentheses. This information is intended for motion editing applications only and does not imply that the data in associated AMC files is actually clipped to the specified range.

:hierarchy Enclosed in a **begin/end** pair, this section describes the tree structure of the kinematic chain. Each line specifies a parent (first entry) and its children (following entries), where parents must be either the **root** or must have been previously referenced as a child. All references are in terms of the bones' **name** fields.

D.1.2 AMC

The AMC file format is as simple as it is impractical to parse. Neither does it contain a field for the sampling rate, nor does its header specify the total number of frames, nor does it give the name of the associated ASF file. Some AMC files contain that information in non-standard comment fields, but we decided to encode the sampling rate and the ASF file in the AMC filename.

After the two header lines, an AMC file gives a list of frames, where each frame is delimited by its frame number. The following lines specify the degrees of freedom for all bones that have associated **dof** fields. The first entry in a line is the bone name, the following entries are the corresponding values for the DOFs, where the sequence is specified by the **order** and

dof fields of the respective bone in the ASF file.

The sequence of bones could be different for every frame. In practice, however, one can assume that all frames specify the bones in the same order, which can be exploited to notably speed up the parsing. Generally, AMC data takes rather long to parse due to the high redundancy and the text-based representation, opposed to a binary format. Readability by a human, not speed, was an issue for the designers of the ASF/AMC format.

Here is an example of a one-frame AMC file associated to the above ASF file.

```
:FULLY-SPECIFIED
:DEGREES
1
root      8.25657  15.4288   6.98449   6.82839  -8.42357  1.70701
lowerback -12.2602   -2.18333  -5.50437
upperback -0.70081   -2.30222  2.4283
thorax    6.47028   -1.33088  5.28671
lowerneck 2.07852   -15.724   -16.5646
upperneck 4.67254   -21.8828  12.1131
head     4.19382  -10.9818   4.95799
rclavicle 0          0
rhumerus -35.355    15.7653   -90.7264
rradius   37.2218
rwrist    -7.61225
rhand     -19.101    20.8962
rfingers  7.12502
rthumb    7.20768   -8.9611
lclavicle 0          0
lhumerus -35.7523   -10.6979  92.1621
lradius   38.1065
lwrist    22.5441
lhand     -8.19849   44.3013
lfingers  7.12502
lthumb    17.7178   73.5141
rfemur    -20.0494   10.5411   21.064
rtibia    41.0307
rfoot     -30.5875   -12.6663
rtoes     -6.70629
lfemur    -20.5304   2.9253    -26.2642
ltibia    42.4012
lfoot     -13.1633   22.5873
ltoes     -13.3539
```

D.1.3 Mapping ASF/AMC Files to Animated Skeletons

In Sect. 1.3.3, we had introduced the notion of an animated skeleton:

$$t \mapsto \underbrace{(J, r, B, (t_b)_{b \in B})}_{\text{ASF}}; \underbrace{(R_r(t), t_r(t), (R_b(t))_{b \in B})}_{\text{AMC}}. \quad (\text{D.1})$$

The set of bones, B , is given by an ASF file's `bonedata:` section. Each bone contributes its distal joint to the joint set, J . Furthermore, J contains the root, r , which is encoded separately in the ASF file. The joint translations, $(t_b)_{b \in B}$, are given for each bone by the `direction` and `length` fields, as described above. These vectors directly refer to the world system, hence one obtains the ASF skeleton's neutral pose (Fig. D.1, left) by appending these vectors to each other according to the hierarchy given by B .

The values of the free parameters given by the AMC file are interpreted as follows. Using the appropriate Euler conversion (Sect. A.6) as specified by the root's `order` field, the root rotation, $R_r(t)$, can be computed for each frame $t \in [1 : T]$. The root translation, $t_r(t)$, is

directly given in world coordinates. Care has to be taken when interpreting the Euler angle data that is given for each bone. First, missing DOFs have to be padded by zero angles at the appropriate positions to obtain full triples of Euler angles. Next, the `axis` values for each bone have to be converted to a rotation using the \overrightarrow{xyz} Euler convention. We denote the axis rotation for bone $b \in B$ by $A_b \in \text{SO}_3$. The time-dependent rotation of bone b as computed from the AMC file is denoted by $R'_b(t)$. Then the bone rotation of the animated skeleton, $R_b(t)$, is obtained by a change of basis as

$$R_b(t) := A_b R'_b(t) A_b^{-1}. \quad (\text{D.2})$$

D.2 The BVH Format

The BVH format is a skeleton-based mocap format that was developed by the mocap services company Biovision. The name BVH stands for “Biovision hierarchical data”. A short documentation can be found at the web page [oW99b]. A BVH file encodes both the skeleton and the free parameters in two blocks, which are titled `HIERARCHY` and `MOTION`. The `HIERARCHY` reflects the tree structure of the kinematic chain by means of a nested structure, where each joint definition is enclosed by braces `{}`, and a joint’s children are listed in the hierarchy level below that joint. The `MOTION` section contains one line of ASCII-coded numbers for each frame, describing the free parameters. BVH files use Euler conventions that are based on a moving reference frame such as \overrightarrow{zxy} , see Sect. A.5.1, and assume that all angles are given in degrees. Lengths are usually specified in centimeters. Here is an excerpt from a BVH file that was taken from the free BVH collection [BVH06].

```
HIERARCHY
ROOT Hips
{
  OFFSET 0.0000 0.0000 0.0000
  CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
  JOINT Chest
  {
    OFFSET -0.1728 10.2870 0.1254
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT Chest2
    {
      OFFSET 0.3229 15.8173 -2.6440
      CHANNELS 3 Zrotation Xrotation Yrotation
      JOINT LeftCollar
      {
        OFFSET -0.0683 16.8828 0.2989
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT LeftShoulder
        {
          OFFSET 16.0110 -3.0750 2.0528
          CHANNELS 3 Zrotation Xrotation Yrotation
          JOINT LeftElbow
          {
            OFFSET 2.6153 -25.4187 1.7231
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT LeftWrist
            {
              OFFSET 2.3595 -22.4741 1.0488
              CHANNELS 3 Zrotation Xrotation Yrotation
              End Site
              {
                OFFSET 1.4871 -13.9919 1.1523
              }
            }
          }
        }
      }
    }
  }
}
```

```

        }
    }
}
JOINT RightCollar
{
    OFFSET -0.0683 16.8828 0.2989
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT RightShoulder
    {
        .
        .
        .
    }
}
.
.
.
}
.
.
.
}
MOTION
Frames: 2
Frame Time: 0.033333
-37.1120 93.3657 37.7112 -3.72 -4.75 1.31 0.76 12.64 1.56 ..... 5.23 -13.70 9.02
-37.1110 93.3769 37.7082 -3.69 -4.70 1.27 0.67 12.70 1.64 ..... 5.22 -13.69 8.98

```

In the following, we explain some of the important keywords and sections of a BVH file.

ROOT	This keyword defines and names a root joint, below which all other joints are assembled. There are BVH files with multiple root joints, enabling more than one character and/or interaction with motion captured objects such as balls to be represented.
JOINT	Defines and names a joint that is not a root.
OFFSET	This is a mandatory field for every joint. It defines the world coordinates of the translation vector from the parent joint to the current joint.
CHANNELS	Declares the number, type, sequence of appearance in the MOTION section, and the Euler convention of the joint's free parameters. For example, Xrotation stands for a rotational degree of freedom about the x axis, while Yposition denotes a translational degree of freedom along the y axis. "Zrotation Xrotation Yrotation" stands for the \overline{zxy} Euler convention.
End Site	Declares an end effector joint, which can only have an OFFSET field.
Frames:	The number of frames, corresponding to the number of data lines that follow below.
Frame Time:	The inverse of the sampling rate, measured in seconds.
Data block	Each data line, corresponding to a single frame, lists the numeric values describing free parameters in the same sequence as the CHANNEL fields appear in the HIERARCHY block.

D.2.1 Mapping BVH Files to Animated Skeletons

The joint set, J , contains all JOINTs appearing in a BVH file, including the root and the End Site joints. The root, r , is given by the BVH’s ROOT field. By the “parent” relation, one obtains the set of bones, B :

$$B := \{(j_1, j_2) \in J \times J \mid j_1 \text{ is parent of } j_2\}. \quad (\text{D.3})$$

The relative joint translations, t_b , are given by the OFFSET vectors. These vectors directly refer to the world system, hence one obtains the BVH skeleton’s neutral pose (Fig. D.1, right) by appending these vectors to each other according to the hierarchy given by B . Note that in the neutral pose, all local coordinate systems at the joints are aligned with the world system.

The root translation, $t_r(t)$, as well as the root rotation, $R_r(t)$, can be directly read off from the data block for each frame $t \in [1 : T]$. Assigning the root rotations to the bones works according to the following scheme: for a bone $b = (j_1, j_2) \in B$, the relative joint rotation $R_b(t)$ is given by the Euler angles corresponding to joint j_1 . Observe that as a consequence of this assignment, bones with a common proximal joint (such as the left/right clavicle and the neck, or the left/right hip joints and the spine) rotate about that common joint as a single, rigid unit.

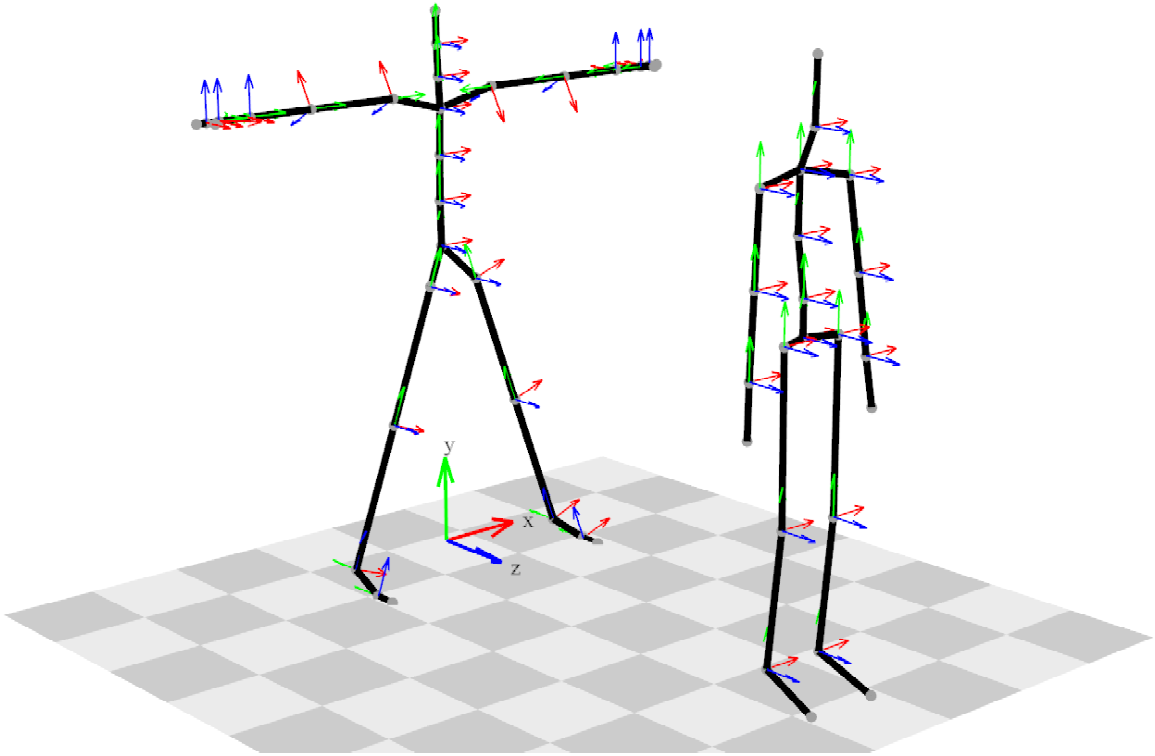


Figure D.1. Neutral poses of the ASF skeleton (left) and the BVH skeleton (right). The respective local coordinate systems are shown at each bone’s proximal joint. For the BVH skeleton, the local coordinate systems are aligned with the world system. The axes are color-coded as red, green, and blue arrows, standing for the x , y , and z axis, respectively.

D.3 Mapping Mocap Data to the Standard Skeleton

Throughout this thesis, we have used the joint-based standard skeleton of Fig. 1.4. Table D.3 explains how the joints of the standard skeleton correspond to the joints of typical ASF and BVH skeletons.

Joint of standard skeleton	Distal joint of ASF bone	Joint of BVH skeleton
root	root	Hips
lhip	lhipjoint	LeftHip
lknee	lfemur	LeftKnee
lankle	ltibia	LeftAnkle
ltoes	ltoes	LeftAnkle_EndSite
rhip	rhipjoint	RightHip
rknee	rfemur	RightKnee
rankle	rtibia	RightAnkle
rtoes	rtoes	RightAnkle_EndSite
belly	lowerback	Chest
chest	upperback	Chest2
neck	thorax	Neck
head	upperneck	Head
headtop	head	Head_EndSite
lclavicle	thorax	LeftCollar
lshoulder	lclavicle	LeftShoulder
lelbow	lhumerus	LeftElbow
lwrist	lwrist	LeftWrist
lfingers	lfingers	LeftWrist_EndSite
rclavicle	thorax	RightCollar
rshoulder	rclavicle	RightShoulder
relbow	rhumerus	RightElbow
rwrist	rwrist	RightWrist
rfingers	rfingers	RightWrist_EndSite

Table D.3. Mapping ASF and BVH skeletons to the joints of the standard skeleton of Fig. 1.4.

D.4 The C3D Format

The C3D format is used by many suppliers of mocap hard- and software to export and exchange raw motion capture data. It is a binary format that is not skeleton-based but instead specifies the 3D trajectories of all markers. A documentation is available at [C3D06], and parsers for Matlab and other programming languages can be found at [CMU03]. One important characteristic of the C3D format is that it allows for additional data streams to be synchronized and stored together with the mocap data. As an example of a typical application, digitized force data from a force plate could be recorded in parallel with the mocap data.

List of Figures

1.1	Muybridge’s electro-photography	5
1.2	Braune and Fischer’s motion capture apparatus	6
1.3	Passive optical motion capture system	8
1.4	Skeletal kinematic chain model	9
1.5	Euclidean motions and local coordinate systems	10
1.6	Forward kinematics	12
1.7	Angle trajectories and 3D trajectories for a ballet motion	15
2.1	Walking vs. jogging	19
2.2	Two walking motions	20
2.3	Point-light displays	21
2.4	Global transformations and motion styles	22
2.5	Two kicking motions	24
2.6	Two jumping motions	25
2.7	Walking motions: curved vs. straight, staircase vs. straight	26
2.8	Rotating arms while walking / while standing	27
2.9	A noisy ballet motion	28
2.10	Quaternion distance measures	30
2.11	Principle of the 3D point cloud distance	31
2.12	Cost matrices for “straight-line walking” vs. “curved walking”	36
2.13	Cost matrices for “rotating arms forwards” with warping paths	37
2.14	Cost matrices for “climbing stairs” vs. “walking”	37
2.15	Cost matrices for two actors lying down on the floor	39
2.16	Cost matrices for two jumping jack motions	39
2.17	Cost matrices for two noisy ballet motions	39
3.1	Examples of relational features	42
3.2	A walking motion	44
3.3	Boolean features for a walking motion	44
3.4	Right foot’s absolute velocity in a walking motion	45
3.5	Details of the right foot in a walking motion	46
3.6	Segmenting a walking motion	48
3.7	F^{walk} -segmentation of $D = D_{\text{walk}}$	48
3.8	F^{r} -segmentation of $D = D_{\text{walk}}$	48
3.9	F^{walk} -segmentation and F^{r} -segmentation of $D = D_{\text{elderlywalk}}$	48
3.10	F^{jump} -segmentation of $D = D_{\text{jump}}$	49

3.11	Right foot kick followed by a left hand punch	50
3.12	Left toes' absolute velocity in a ballet motion	55
3.13	Right wrist's upward velocity in a jumping motion	56
3.14	Robust relational features	58
3.15	Robust thresholding: feature values	59
3.16	Robust thresholding: histograms of segment lengths	60
3.17	First three principal components of training database	61
3.18	Factor loadings for first two principal components	61
3.19	Distance signals with strongest factor loadings	61
3.20	3D plots of first three principal components	62
3.21	Relational feature matrices for "walking" and "jogging"	68
3.22	Cost matrices for "walking" and "rotating arms" based on relational features	70
3.23	Cost matrices for "lying down", "jumping jack", and "ballet" based on relational features	70
3.24	Cost matrices for "climbing stairs" based on relational features	70
4.1	Exact hits for walking	73
4.2	Exact hits for jumping	73
4.3	Fuzzy hits for walking	74
4.4	Fuzzy hits for jumping	74
4.5	Adaptive fuzzy hits for walking	75
4.6	Adaptive fuzzy hits for jumping	75
4.7	Overview of the retrieval process based on the query-by-example paradigm. .	80
4.8	Preprocessing and querying	80
4.9	Histogram of inverted list lengths	81
4.10	DTW-based refinement of hits	83
4.11	Hits for "punching" and "squatting"	85
4.12	Hits for "kicking"	86
4.13	Self-similarity matrix for a gymnastics motion	91
4.14	Principle of motion retrieval based on self-similarity	91
5.1	Four cartwheels and feature matrices	94
5.2	Average MT and unwarped average MT	95
5.3	Class MT and quantized class MT for 'CartwheelLeft'	96
5.4	Class MTs for 'CartwheelLeft' and 'JumpingJack'	97
5.5	Class MTs for 'walking sideways' and 'rotating arms'	98
5.6	Distance functions Δ_C for quantized MT and MT	100
5.7	Distance functions for a gymnastics motion	100
5.8	Cost matrices with warping paths for a gymnastics motion	100
A.1	Geometric interpretation of Euler angles	129
A.2	Gimbal assembly for two-axis rotation orders	131
A.3	Gimbal lock	132
A.4	Closed path on S^3	135
A.5	Euler angle trajectories corresponding to a closed path on S^3	135
A.6	Euler angles and quaternions for the right knee angle in a dance motion . . .	136
A.7	Plot of the two-argument arctan2 function	141

B.1	Spherical averaging vs. orientation filter	151
B.2	Filtering a synthetic quaternion signal	153
B.3	Filtering a quaternion signal derived from a ballet motion	153
B.4	Filtering a ballet motion	154
B.5	Left toes' absolute velocity in a ballet motion after filtering	154
D.1	Neutral pose of ASF and BVH skeletons	166

List of Tables

3.1	Feature set E	67
3.2	Feature set F	67
4.1	Feature computation and index construction	84
4.2	Statistics on 10,000 random queries in I_u^{180}	85
5.1	Ten motion classes from \mathcal{D}^{MC}	95
5.2	Retrieval results on \mathcal{D}^{MCE} , part 1	103
5.3	Retrieval results on \mathcal{D}^{MCE} , part 2	104
5.4	Retrieval results on \mathcal{D}_{210}	106
5.5	Comparison of MT-based matching to baseline methods	108
5.6	Comparison of adaptive fuzzy retrieval and MT-based retrieval	109
A.1	Qualitative comparison of parametrizations of $SO(\mathbb{R}^3)$	115
A.2	Euler-to-matrix conversion for twelve Euler conventions	138
A.3	Euler-to-quaternion conversion for twelve Euler conventions	140
A.4	Matrix-to-Euler conversion for twelve Euler conventions.	142
A.5	Quaternion-to-Euler conversion for twelve Euler conventions.	143
D.3	Mapping ASF and BVH skeletons to our standard skeleton	167

Bibliography

- [ACCO05] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: pose selection and illustration. *ACM Trans. Graph.*, 24(3):667–676, 2005.
- [AFO03] Okan Arikan, David A. Forsyth, and James F. O’Brien. Motion synthesis from annotations. *ACM Trans. Graph.*, 22(3):402–408, 2003.
- [AHB87] K. S. Arun, Thomas S. Huang, and Steven D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.
- [All00] Jonathan Allday. *Apollo in Perspective: Spaceflight then and now*. Institute of Physics Publishing, 2000.
- [ALP04] Yeuhi Abe, C. Karen Liu, and Zoran Popović. Momentum-based parameterization of dynamic character motion. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2004)*, pages 173–182, New York, NY, USA, 2004. ACM Press.
- [Asc06] Ascension. Magnetic motion capture. <http://www.ascension-tech.com/products/motionstar.php>, 2006.
- [BCMS01] Alessandro Bissacco, Alessandro Chiuso, Yi Ma, and Stefano Soatto. Recognition of human gaits. In *Proc. IEEE CVPR*, pages 401–417. IEEE, December 2001.
- [BDC02] Chiraz BenAbdelkader, Larry S. Davis, and Ross Cutler. Motion-based recognition of people in eigengait space. In *Proc. 5th IEEE Intl. Conf. Automatic Face and Gesture Recognition (FGR 2002)*, pages 267–274, 2002.
- [BF91] Wilhelm Braune and Otto Fischer. Die Bewegung des Kniegelenks nach einer neuen Methode am lebenden Menschen gemessen. *Abhandl. d. Math.-Phys. Cl. d. k. Sächs. Gesellsch. Wissensch.*, XVII(II), 1891.
- [BF94] Wilhelm Braune and Otto Fischer. Der Gang des Menschen, 1. Teil: Versuche am unbelasteten und belasteten Menschen. *Abhandl. d. Math.-Phys. Cl. d. k. Sächs. Gesellsch. Wissensch.*, XXI, 1894.
- [BF87] Wilhelm Braune and Otto Fischer. *Determination of the moments of inertia of the human body and its limbs*. Springer, 1987. Translated by Paul Maquet and Ronald Furlong.

- [BF01] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph.*, 20(2):95–126, 2001.
- [BH00] Matthew Brand and Aaron Hertzmann. Style machines. In *Proc. ACM SIGGRAPH 2000*, Computer Graphics Proc., pages 183–192. ACM Press, 2000.
- [BPW93] Norman I. Badler, Cary B. Phillips, and Bonnie L. Webber. *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press, 1993.
- [Bre97] Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *Proc. CVPR 1997*, page 568, Washington, DC, USA, 1997. IEEE Computer Society.
- [BSP⁺04] Jernej Barbic, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K. Hodgins, and Nancy S. Pollard. Segmenting motion capture data into distinct behaviors. In *GI '04: Proc. Graphics Interface*, pages 185–194. Canadian Human-Computer Communications Society, 2004.
- [BVH06] BVHFILES. Free BVH mocap archive. <http://www.bvhfiles.com>, 2006.
- [BW95] Armin Bruderlin and Lance Williams. Motion signal processing. In *Proc. ACM SIGGRAPH 1995*, Computer Graphics Proc., pages 97–104. ACM Press, 1995.
- [C3D06] C3D.org. C3D mocap format. <http://www.c3d.org>, 2006.
- [Car96] Stefan Carlsson. Combinatorial geometry for shape representation and indexing. In *Object Representation in Computer Vision*, pages 53–78, 1996.
- [Car99] Stefan Carlsson. Order structure, correspondence, and shape based categories. In *Shape, Contour and Grouping in Computer Vision*, pages 58–71. Springer, 1999.
- [CB95] Lee W. Campbell and Aaron F. Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, pages 624–630, 1995.
- [CH05] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.
- [CK04] Michael Clausen and Frank Kurth. A unified approach to content-based and fault tolerant music recognition. *IEEE Trans. Multimedia*, 6(5):717–731, 2004.
- [CKK03] Michael Clausen, Heiko Körner, and Frank Kurth. An efficient indexing and search technique for multimedia databases. In *SIGIR 2003 Workshop on Multimedia Retrieval*, 2003.
- [CMU03] CMU. Carnegie-Mellon Mocap Database. <http://mocap.cs.cmu.edu>, 2003.
- [CVKG03] Michalis Cardle, Stephen B. Vlachos, Eamonn Keogh, and Dimitrios Gunopulos. Fast motion capture matching with replicated motion editing. *ACM SIGGRAPH 2003, Sketches and Applications*, 2003.

- [dATS06] Edilson de Aguiar, Christian Theobalt, and Hans-Peter Seidel. Automatic learning of articulated skeletons from 3D marker trajectories. In *Proc. Intl. Symposium on Visual Computing (ISVC 2006)*, to appear, 2006.
- [DB97] James W. Davis and Aaron F. Bobick. The representation and recognition of human movement using temporal templates. In *Proc. CVPR '97*, pages 928–934, Washington, DC, USA, 1997. IEEE Computer Society.
- [Dem06] Bastian Demuth. Beiträge zu einem Such- und Klassifikationssystem für Bewegungsdaten. Master's thesis, University of Bonn, 2006.
- [DG03] James W. Davis and Hui Gao. An expressive three-mode principal components model of human action style. *Image Vision Comput.*, 21(11):1001–1016, 2003.
- [DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley Interscience, second edition, 2000.
- [DRME06a] Bastian Demuth, Tido Röder, Meinard Müller, and Bernhard Eberhardt. An information retrieval system for motion capture data. In *Proc. 28th European Conference on Information Retrieval (ECIR 2006)*, volume 3936 of *LNCS*, pages 373–384. Springer, 2006.
- [DRME06b] Bastian Demuth, Tido Röder, Meinard Müller, and Bernhard Eberhardt. Web site accompanying the ECIR 2006 paper “An information retrieval system for motion capture data”. http://www-mmdb.iai.uni-bonn.de/projects/mocap/RetrievalGUI_results/RetrievalGUI.html, 2006.
- [Fau01] Olivier Faugeras. *Three-dimensional Computer Vision—A Geometric Viewpoint*. MIT Press, 2001.
- [FF05] Kevin Forbes and Eugene Fiume. An efficient search algorithm for motion data using weighted PCA. In *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–76. ACM Press, 2005.
- [Fio01] Paul D. Fiore. Efficient linear solution of exterior orientation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2):140–148, 2001.
- [Fis99] Otto Fischer. Der Gang des Menschen, 2. Teil: Die Bewegung des Gesamtschwerpunktes und die äusseren Kräfte. *Abhandl. d. Math.-Phys. Cl. d. k. Sächs. Gesellsch. Wissensch.*, XXV, 1899.
- [FLPJ04] P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang C. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Trans. Med. Imaging*, 23(8):995–1005, 2004.
- [FMJ02] Ajo Fod, Maja J. Mataric, and Odest Chadwicke Jenkins. Automated derivation of primitives for movement classification. *Auton. Robots*, 12(1):39–54, 2002.
- [För98] Wolfgang Förstner. Skriptum zur Vorlesung Photogrammetrie. University of Bonn, Dept. of Photogrammetry, 1998.

- [GG04] Richard D. Green and Ling Guan. Quantifying and recognizing human movement patterns from monocular video images: Part I. *IEEE Trans. Circuits and Systems for Video Technology*, 14(2):179–190, February 2004.
- [Gol02] E. Bruce Goldstein. *Wahrnehmungspsychologie*. Spektrum Akademischer Verlag, Heidelberg, Berlin, second edition, 2002.
- [Gor83] Richard L. Gorsuch. *Factor analysis*. Erlbaum, Hillsdale, NJ, 1983.
- [GP00] M.A. Giese and T. Poggio. Morphable models for the analysis and synthesis of complex motion patterns. *IJCV*, 38(1):59–73, 2000.
- [Gra98] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *J. Graphics Tools*, 3(3):29–48, 1998.
- [Gue04] Ann Hutchinson Guest. *Labanotation: the system of analyzing and recording movement*. Routledge, New York, 2004.
- [GY05] Tong Guan and Yee-Hong Yang. Motion similarity analysis and evaluation of motion capture data. Technical Report TR05-11, University of Alberta, Edmonton, Alberta, Canada, 2005.
- [Gyp06] Gypsy. Mechanical motion capture. <http://www.animazoo.com>, 2006.
- [Har78] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proc. IEEE*, 66(1):51–83, 1978.
- [HN01] Kozaburo Hachimura and Minako Nakamura. Method of generating coded description of human body motion from motion-captured data. In *Proc. 10th IEEE Intl. Workshop on Robot and Human Interactive Communication*, 2001.
- [HP03] Michael Hofer and Helmut Pottmann. Orientierung von Laserscanner-Punktwolken. *Vermessung & Geoinformation*, 91:297–306, 2003.
- [HPP05] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Trans. Graph.*, 24(3):1082–1089, 2005.
- [Joh75] Gunnar Johansson. Visual motion perception. *Scientific American*, 232:76–89, 1975.
- [Joh03] Michael P. Johnson. *Exploiting Quaternions to Support Expressive Interactive Character Motion*. PhD thesis, Massachusetts Institute of Technology, 2003.
- [Kar77] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Math*, 30(5):509–541, 1977.
- [KB96] Hyeongseok Ko and Norman I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, 1996.
- [KC77] Lynn T. Kozlowski and James E. Cutting. Recognizing the sex of a walker from a dynamic point-light display. *Perception and Psychophysics*, 21:575–580, 1977.

- [Keo02] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proc. 28th VLDB Conf., Hong Kong*, pages 406–417, 2002.
- [KG03] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224. Eurographics Association, 2003.
- [KG04] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568, 2004.
- [KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, 2002.
- [Koe92] Max Koecher. *Lineare Algebra und analytische Geometrie*. Springer, 1992.
- [KP06] Paul G. Kry and Dinesh K. Pai. Interaction capture and synthesis. *ACM Trans. Graph.*, 25(3):872–880, 2006.
- [KPZ⁺04] Eamonn J. Keogh, Themis Palpanas, Victor B. Zordan, Dimitrios Gunopulos, and Marc Cardle. Indexing large human-motion databases. In *Proc. 30th VLDB Conf., Toronto*, pages 780–791, 2004.
- [Krü06] Björn Krüger. Dynamik basiertes Morphing von Bewegungsdaten. Master’s thesis, University of Bonn, 2006.
- [Kub03] Klaus D. Kubinger. On artificial results due to using factor analysis for dichotomous variables. *Psychology Science*, 45(1):106–110, 2003.
- [LCR⁺02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive Control of Avatars Animated with Human Motion Data. In *Proc. SIGGRAPH 2002*, pages 491–500. ACM Press, 2002.
- [LE04] Chan-Su Lee and Ahmed Elgammal. Gait style and gait content: Bilinear models for gait recognition using gait re-sampling. In *Proc. IEEE Intl. Conf. Automatic Face and Gesture Recognition (FGR 2004)*, pages 147–152. IEEE Computer Society, 2004.
- [Lev66] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [LHP05] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.*, 24(3):1071–1081, 2005.
- [LKB04] Mark Last, Abraham Kandel, and Horst Bunke, editors. *Data Mining In Time Series Databases*. World Scientific, 2004.
- [LLL92] M. A. Lafortune, C. Lambert, and M. Lake. Skin marker displacement at the knee joint. In *Proc. 2nd North American Congress on Biomechanics*, Chicago, 1992.

- [LP03] Qiang Liu and Edmond Prakash. The parameterization of joint rotation with the unit quaternion. In *Proc. 7th Intl. Conf. Digital Image Computing: Techniques and Applications (DICTA 2003)*, pages 409–418, 2003.
- [LS02] Jehee Lee and Sung Yong Shin. General construction of time-domain filters for orientation data. *IEEE Trans. Visualization and Computer Graphics*, 8(2):119–128, 2002.
- [LWS02] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proc. ACM SIGGRAPH 2002*, pages 465–472. ACM Press, 2002.
- [LZWM05] Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. A system for analyzing and indexing human-motion databases. In *Proc. 2005 ACM SIGMOD Intl. Conf. on Management of Data*, pages 924–926. ACM Press, 2005.
- [MCA06] Lars Mündermann, Stefano Corazza, and Thomas P Andriacchi. The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *J. Neuroengineering Rehabil.*, 3(6), March 2006.
- [MR06a] Meinard Müller and Tido Röder. Web site accompanying the SCA 2006 paper “Motion templates for automatic classification and retrieval of motion capture data”. http://www-mmdb.iai.uni-bonn.de/projects/mocap/SCA2006_results/SCA2006.html, 2006.
- [MR06b] Meinard Müller and Tido Röder. Motion templates for automatic classification and retrieval of motion capture data. In *Proc. 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)*. Eurographics Association, 2006.
- [MRC05a] Meinard Müller, Tido Röder, and Michael Clausen. Web site accompanying the SIGGRAPH 2005 paper “Efficient content-based retrieval of motion capture data”. http://www-mmdb.iai.uni-bonn.de/projects/mocap/SIGGRAPH2005_results/SIGGRAPH2005.html, 2005.
- [MRC05b] Meinard Müller, Tido Röder, and Michael Clausen. Efficient content-based retrieval of motion capture data. *ACM Trans. Graph.*, 24(3):677–685, 2005.
- [MRC05c] Meinard Müller, Tido Röder, and Michael Clausen. Efficient indexing and retrieval of motion capture data based on adaptive segmentation. In *Proc. Fourth International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2005.
- [Muy87] Eadweard Muybridge. *Animal Locomotion: An Electro-Photographic Investigation of Consecutive Phases of Animal Movement*. University of Pennsylvania, Philadelphia, 1887.
- [Muy01] Eadweard Muybridge. *The Human Figure in Motion: An Electro-Photographic Investigation of Consecutive Phases of Muscular Actions*. Chapman & Hall, London, 1901.

- [NF04] Michael Neff and Eugene Fiume. Methods for exploring expressive stance. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2004)*, pages 49–58. ACM Press, 2004.
- [NF05] Michael Neff and Eugene Fiume. AER: aesthetic exploration and refinement for expressive character animation. In *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2005)*, pages 161–170. ACM Press, 2005.
- [OBBH00] James F. O’Brien, Robert Bodenheimer, Gabriel Brostow, and Jessica K. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Graphics Interface*, pages 53–60, 2000.
- [oW99a] University of Wisconsin. ASF/AMC mocap format. <http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html>, 1999.
- [oW99b] University of Wisconsin. BVH mocap format. <http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html>, 1999.
- [PB02] Katherine Pullen and Chris Bregler. Motion capture assisted animation: texturing and synthesis. In *Proc. SIGGRAPH 2002*, pages 501–508. ACM Press, 2002.
- [Pea01] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.
- [Pop99] Zoran Popović. *Motion Transformation by Physically Based Spactime Optimization*. PhD thesis, Carnegie Mellon University, 1999.
- [Por95] Ian R. Porteous. *Clifford Algebras and the Classical Groups*. Cambridge Studies in Advanced Mathematics (50), 1995.
- [Pul02] Katherine Pullen. *Motion capture assisted animation: texturing and synthesis*. PhD thesis, Stanford University, 2002.
- [RCB98] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.
- [Rei96] Christoph Reinschmidt. *Three-dimensional tibiocalcaneal and tibiofemoral kinematics during human locomotion—measured with external and bone markers*. PhD thesis, Department of Medical Science, University of Calgary, 1996.
- [RF81] S. Runeson and G. Frykholm. Visual perception of lifted weights. *J. Exp. Psych.: Human Perception and Performance*, 7:733–740, 1981.
- [RF03] Deva Ramanan and David A. Forsyth. Automatic annotation of everyday movements. In *Advances in Neural Information Processing Systems 16*, 2003.
- [Rib06] Andreas Ribbrock. *Effiziente Algorithmen und Datenstrukturen zur inhaltsbasierten Suche in Audio- und 3D-Moleküldaten*. PhD thesis, University of Bonn, 2006.

- [Ric99] James G. Richards. The measurement of human motion: A comparison of commercially available systems. *Human Movement Science*, 18:589–602, 1999.
- [RJ93] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.
- [RKS⁺05] Bodo Rosenhahn, Uwe G. Kersting, Andrew W. Smith, Jason K. Gurney, Thomas Brox, and Reinhard Klette. A system for marker-less human motion estimation. In *DAGM-Symposium*, pages 230–237, 2005.
- [Rod40] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d’un système solide dans l’espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. *Journal de Mathématiques Pures et Appliquées*, 5:380–440, 1840.
- [RPE⁺05] Liu Ren, Alton Patrick, Alexei A. Efros, Jessica K. Hodgins, and James M. Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. Graph.*, 24(3):1090–1097, 2005.
- [SC02] Josephine Sullivan and Stefan Carlsson. Recognizing and tracking human action. In *Proc. ECCV ’02, Part I*, pages 629–644. Springer, 2002.
- [Sch97] Bernd Schmidt. Skriptum zur Vorlesung Lineare Algebra 1/2. University of Bonn, Dept. of Mathematics, 1997.
- [Sha06] ShapeWrap. Mechanical motion capture. <http://www.measurand.com>, 2006.
- [Sho85] Ken Shoemake. Animating rotations with quaternion curves. In *Proc. SIGGRAPH 1985*, pages 245–254. ACM Press, 1985.
- [Sho94] Ken Shoemake. Euler angle conversion. In Paul S. Heckbert, editor, *Graphics Gems IV*, chapter III.5. Academic Press Professional, Inc., San Diego, CA, USA, 1994.
- [SKK04] Yasuhiko Sakamoto, Shigeru Kuriyama, and Toyohisa Kaneko. Motion map: image-based retrieval and segmentation of motion data. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 259–266. ACM Press, 2004.
- [SLZ⁺03] Nicu Sebe, Michael S. Lew, Xiang Sean Zhou, Thomas S. Huang, and Erwin M. Bakker. The state of the art in image and video retrieval. In *CIVR*, volume 2728 of *LNCS*, pages 1–8. Springer, 2003.
- [Stu94] David J. Sturman. SIGGRAPH course 9: Character motion systems. In *Proc. SIGGRAPH ’94*. ACM SIGGRAPH, 1994.
- [Tro02] Nikolaus F. Troje. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *J. Vis.*, 2(5):371–387, 9 2002.
- [TWL05] Nikolaus F. Troje, Cord Westhoff, and Mikhail Lavrov. Person identification from biological motion: Effects of structural and kinematic cues. *Perception & Psychophysics*, 67:667–675, 2005.

- [UAT95] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Proc. ACM SIGGRAPH 1995*, pages 91–96. ACM Press, 1995.
- [Ume91] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.
- [VHGK06] Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15(1):1–20, 2006.
- [Vic06] Vicon. Optical motion capture. <http://www.vicon.com>, 2006.
- [vL95] Rudolph von Laban. *Kinetografie. Labanotation. Einführung in die Grundbegriffe der Bewegungs- und Tanzschrift*. Noetzel, Wilhelmshaven, 1995.
- [WB99] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):884–900, 1999.
- [WCYL03] Ming-Yang Wu, Shih-Pin Chao, Shi-Nine Yang, and Hsin-Chih Lin. Content-based retrieval for human motion data. In *16th IPPR Conf. on Computer Vision, Graphics, and Image Processing*, pages 605–612, 2003.
- [WMB99] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, 1999.
- [WP95] Andrew Witkin and Zoran Popović. Motion warping. In *Proc. ACM SIGGRAPH 95*, Computer Graphics Proc., pages 105–108. ACM Press/ACM SIGGRAPH, 1995.
- [YSLG05] Tao Yu, Xiaojie Shen, Qilei Li, and Weidong Geng. Motion retrieval based on movement notation language. *Comp. Anim. Virtual Worlds*, 16(3–4):273–282, 2005.
- [Zat98] Vladimir M. Zatsiorsky. *Kinematics of Human Motion*. Human Kinetics, 1998.
- [ZMCF05] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701, 2005.

Index

- S^3 , 118
- RP^3 , 120
- $SO(\mathbb{R}^3)$, 113
- artifacts, 27
- ASF/AMC, 159
- bone, [11](#)
- BVH, 164
- C3D, 167
- calibration, 7
- Cardan ring, 128
- center of mass, 56
- chronophotography, 5
- class template, 95
- classification, 101
- coordinate frame
 - fixed, 127
 - moving, 127
- coordinate system
 - affine, 9
 - local, [9](#)
 - root, 12
 - world, [10](#)
- coordinates
 - local, [10](#)
- cost matrix, [34](#)
- cost measure, 28, 99
- database
 - evaluation, 96
 - motion class, 96
 - training, 96
- degree of freedom, *see* DOF
- deletion, 87
- DOF, 9, [14](#)
 - AMC file, 162
 - ASF file, 162
 - BVH file, 165
 - rotational, 131
- DTW, 22
 - subsequence, 27, 99
- dynamic time warping, *see* DTW
- dynamics, 56
- dyneme, 24, 110
- electro-photography, 5
- Euclidean motion, 10
- Euler angle, 123
 - convention, 124
 - inversion, 128
 - reference frame, 125
- expressive, 24
- factor, 62
 - loadings, 62
- false negative, 25, 82
- false positive, 25, 82
- feature
 - boolean, 41
 - function, 43
 - global, 79
 - generic relational, 43, 50
 - $F_{\theta,\text{angle}}$, 52
 - $F_{\theta,\text{fast}}$, 53
 - $F_{\theta,\text{move}}$, 55
 - $F_{\theta,\text{nmove}}$, 54
 - $F_{\theta,\text{nplane}}$, 51
 - $F_{\theta,\text{plane}}$, 51
 - $F_{\theta,\text{touch}}$, 52
 - matrix, [68](#)
 - relational, 2, 41
 - selection, 80
 - sequence, 46
 - value, 43
 - vector, 43
- filter gain, 149, [150](#)
- force, 56

- forward kinematics, 9, 12
- frame, 5
- Gaussian lowpass filter, 54
- gimbal, 128
- gimbal lock, 120, 125, 131
- hit, 90
 - adaptive fuzzy, 73, 74, 79
 - exact, 71, 76
 - false negative, 82
 - false positive, 82
 - fuzzy, 72, 77
- index, 76, 82, 105
- insertion, 87
- intrinsic mean, 146
- invariance, 20, 22
- inverted list, 75, 105
- joint, 11
 - distal, 11
 - end effector, 11
 - prismatic, 9
 - proximal, 11
 - revolute, 9
 - root, 12
 - rotating, 11
 - shoulder, 16
 - space, 14
 - torque, 56
- Karcher mean, 146
- keyframe, 102
- keyframing, 1
- kinematic chain, 8, 11
 - ASF, 163
 - BVH, 164
 - open, 8
 - skeletal, 14
- Labanotation, 89
- latent variable, 62
- load, 62
- markers
 - active, 6
 - bone-implanted, 16
 - magnetic, 7
 - optical, 6
 - passive, 6
 - placement, 7, 16
 - shifting, 16, 17
 - skin-attached, 16
- match web, 85, 90, 111
- metadata, 2
- morphing and blending, 2
- motion
 - analysis, 1, 2
 - artifacts, 27
 - aspect, 20
 - classification, 101
 - content, 23, 88
 - data stream, 5
 - Euclidean, 10
 - fingerprint, 3, 47
 - noise, 27
 - retrieval, 71, 99
 - reuse, 2
 - style, 22
 - synthesis, 1
 - texton, 23
 - texture, 23
 - unnatural, 28
- motion analysis
 - biomechanical, 6, 16
- motion capture
 - accuracy, 17
 - cleaning, 8
 - magnetic, 7
 - marker placement, 7, 16
 - mechanical, 7
 - optical, 6
 - skeletal fitting, 8
 - technology, 5
- motion capture data, 5
 - ASF/AMC format, 159
 - BVH format, 164
 - C3D format, 167
 - recording, 7
- motion class, 96
- motion template, 93
 - class, 95
 - quantized, 99
 - reference, 94
- moveme, 24

- MT, [93](#)
- multiplication order, 127
- noise, 27, 53
- orientation filter, 145, 148, [150](#)
 - mask, 150
- padding, 54
- parameter, 9
 - free, [14](#)
 - AMC, 163
 - BVH, 164
 - skeletal, [14](#)
 - ASF, 163
 - BVH, 164
- parametrization, 113
- PCA, 62
- perception, 20
- phoneme, 24
- point-light display, 20
- pose, [5](#)
 - distance
 - 3D point cloud, 31
 - quaternion-based, 29
 - space, [5](#)
- Procrustes, 32
- projective space, 120
- quaternion, [118](#)
 - conjugate, [118](#)
 - exponential map, [122](#)
 - logarithm, [123](#)
 - norm, [118](#)
 - path continuity, 121
 - pure, [118](#)
 - sequence, 121
 - unit, [118](#)
- query
 - admissible fuzzy, [75](#), 105
 - by example, 79
 - exact, [71](#)
 - fuzzy, [72](#)
 - mismatch, 75
- range of motion, 7, 89
- ranking, 22, 81, 82
- relational feature, 41
- relevance, 26
- retrieval, 71, 99
 - content-based, 2
- Rodrigues matrix, 116
- root, 8, [11](#)
- rotation
 - angle, 116
 - axis, 114, 116
 - basic, 115
 - matrix, 114
 - order, 127
 - three-axis, 124
 - two-axis, 127
 - plane, 114
 - reference frame, 125
 - relative joint, [11](#)
 - root, [11](#)
- rotoscopy, 6
- score function, 28
- segment, 46
 - boundaries, 76, 81, 105
 - length, 76
 - transitory, 47, 72
- segmentation, 2
- self-similarity matrix, 90
- semantic gap, 25, 41
- separation quotient, 107
- similarity, 19
 - aspect of, 20
 - emotional, 23
 - mood, 23
 - logical, 19, 24
 - measure, 28
 - numerical, 20
 - partial, 26
 - perceptual, 31
 - subsequence, 27
- similarity measure
 - global, 34
 - local, 26, 28
 - semi-local, 33, 88, 89
- similarity, motion, 2
- skeleton, 14, 16
 - animated, [14](#)
 - ASF/AMC, 163
 - BVH, 166

- standard, 8, 167
- skin shift, 16, 17
- slerp, 120
- spherical weighted average, 147
- style machines, 23
- style translation, 23

- T-pose, 7
- take, 7
- temporal templates, 21
- test plane, 43
- tetrachoric correlation, 65
- thresholding
 - hysteresis, 58
 - robust, 58
- training example
 - positive, 108
- trajectory
 - angle, 14
 - joint, 5
- translation
 - relative joint, 11
 - root, 11

- variations
 - spatial, 25
 - spatio-temporal, 25
 - temporal, 25, 46
- velocity, 53
- verbs and adverbs, 23

- warping path, 34
- wobbling mass, 16